NAVAL POSTGRADUATE SCHOOL
Monterey, California

UNITED STATES NAVY • NAVAL POSTGRADUATE SCHOOL •

THESIS

PERSONAL COMPUTER and WORKSTATION OPERATING SYSTEMS TUTORIAL

by

Charles E. Frame Jr.

March, 1994

Thesis Advisor: Norman F. Schneidewind

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1994	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE PERSONAL COMPUTER and WORKSTATION OPERATING SYSTEMS TUTORIAL	5. FUNDING NUMBERS
6. AUTHOR Charles E. Frame Jr.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORIN G AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE A

13. ABSTRACT *(maximum 200 words)*

This thesis is a review and analysis of personal computer and workstation operating systems. The emphasis is placed on UNIX, MS DOS, MS Windows and OS/2 operating systems. UNIX is covered under the U.S. Government POSIX standard, which dictates its use when practical. MS DOS is the most used operating system worldwide. OS/2 was developed to combat some of the shortcomings of MS DOS. Each operating system which is discussed has a design philosophy that fulfills specific user's needs. UNIX was designed for many users sharing a computer system. MS DOS, MS Windows and OS/2 are designed as single user computer systems. All of these operating systems are in use at the Naval Postgraduate School.

All of the operating systems are discussed with regard to their: history of development, process management, file system, input and output system, user interface, network capabilities, and advantages and disadvantages. UNIX has a section devoted to the POSIX standard and MS DOS has a section devoted to Windows 3.1. Apple Corporation's System 7 is mentioned throughout the text, but is not covered in detail.

14. SUBJECT TERMS Personal Computer and Workstation Operating Systems : UNIX, MS DOS, OS/2.			15. NUMBER OF PAGES 155
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Personal Computer and Workstation Operating Systems Tutorial

by

Charles E. Frame Jr.
Lieutenant, United States Navy
B.S., Auburn University, 1986

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
March 1994

Author: Charles E. Frame Jr.

Charles E. Frame Jr.

Approved by: [illegible]

Norman F. Schneidewind, Thesis Advisor

[illegible]

Myung W. Suh, Second Reader

[illegible]

David R. Whipple, Chairman
Department of Systems Management

ABSTRACT

This thesis is a review and analysis of personal computer and workstation operating systems. The emphasis is placed on UNIX, MS DOS, MS Windows and OS/2 operating systems. UNIX is covered under the U.S. Government POSIX standard, which dictates its use when practical. MS DOS is the most used operating system worldwide. OS/2 was developed to combat some of the shortcomings of MS DOS. Each operating system which is discussed has a design philosophy that fulfills specific user's needs. UNIX was designed for many users sharing a computer system. MS DOS, MS Windows and OS/2 are designed as single user computer systems. All of these operating systems are in use at the Naval Postgraduate School.

All of the operating systems are discussed with regard to their: history of development, process management, file system, input and output system, user interface, network capabilities, and advantages and disadvantages. UNIX has a section devoted to the POSIX standard and MS DOS has a section devoted to Windows 3.1. Apple Corporation's System 7 is mentioned throughout the text, but is not covered in detail.

TABLE OF CONTENTS

I. OPERATING SYSTEM BACKGROUND

A. INTRODUCTION

Today's computer customer is faced with a wide array of decisions. Changes in hardware and software occur so frequently that a dedicated effort is required to maintain currency. An important part of the computer hardware is the Central Processing Unit (CPU). Since 1971 Intel Corporation has released eight major CPU designs, each an improvement over the previous design. Intel and its competitor Motorola have both released new CPU designs in 1993.

The most important piece of system software to the microcomputer is the operating system. The operating system is a group of programs that allocates resources and manages execution of user programs. It also acts as an interface between the user and the computer hardware. In 1993 the three major operating system vendors for the personal computer (PC) all released new versions of their systems. Apple released System 7.1 for the Macintosh. IBM and Microsoft released OS/2 2.1 and MS DOS 6.2 respectively.

The emphasis in this thesis is on operating systems. The purpose is to provide a tutorial that promotes informed decisions about PC operating systems. In this introductory section a brief look into hardware will include: CPU's, computer memory, and some network communication devices. This is followed by some background information on operating system software.

Following this Background chapter, three PC operating systems are discussed in the order of their chronological development. These systems are: UNIX, MS DOS, and OS/2. UNIX is chosen because it is covered by the Government POSIX standard for operating systems. It also forms the basis for parts of MS DOS and OS/2. A brief mention of Microsoft's LAN Manager is also presented along with IBM's LAN Server because they are both based on the OS/2 operating system.

The Macintosh System 7 is not given close scrutiny since Apple has integrated much of this operating system into hardware. Microsoft's WindowsNT is also not fully discussed since it is not considered a PC operating system. All of these systems are summarized in the final chapter to facilitate a an operating system comparison.

B. CENTRAL PROCESSING UNITS

The current design of the computer is based on the von Neumann architecture that consists of the three key concepts: [Ref. 1:p. 204]

* Data and instructions are stored in a single read- write memory (Random Access Memory or RAM)
* The contents of the memory is addressable without regard to contents of the memory.
* Execution of the instructions are sequential unless specified otherwise.

Avoiding details of registers that hold addresses, data and instructions, consider only that a program is read from a fixed disk to a RAM area where it is executed by the CPU. The program consists of instructions that perform operations on data. The CPU will perform two basic steps. [Ref. 2:p. 82] First the CPU will fetch an instruction and then the CPU will execute the instruction. Together a fetch and an execute form a CPU cycle.

To join the CPU to memory and input/output devices a highway called a system bus is used. [Ref. 2:p. 232] The system bus is actually three basic buses, one for control (timing), one for addressing memory, and one for data transfer. It is the control bus that provides the speed of the CPU measured in megahertz (MHZ) or millions of cycles per second. For example, a CPU operating at 25 MHZ is performing 25 million cycles per second. This is a measure of the computers raw internal speed.

Buses are extremely important to utilize the potential power of the CPU. There are actually more than just the three basic busses mentioned. [Ref. 3:p. 70] A local bus that runs at the same speed of the CPU may be used with a video controller device. An intermediate bus may be connected to the CPU's data bus through a buffer controller for data operations. A Small Computer System Interface (SCSI) often used with fixed disk drives and CD-ROM drives may be considered a type of bus. SCSI requires a controller that fits into a computers bus socket.

In the jargon of computers two types of buses seem to dominate. [Ref. 3:p. 72] The "ISA" (Industry Standard Architecture) bus is the older model used to support a 16 bit bus architecture. The "EISA" (Extended Industry Standard Architecture) bus is the newer model that supports a 32 bit bus architecture. Other types of buses exist, but this is merely a fundamental background to aid in the understanding of 16 and 32 bit architectures for computers.

The CPU communicates along a bus at some clock rate measured in megahertz. To measure the work being performed by the CPU, three performance indicators are traditionally used. These three indicators are: Millions of Instructions per Second (MIPS), Millions of Floating Point Operations per Second (MFLOPS), and Throughput. MIPS is the most common measure, but for more mathematically intensive operations scientists may prefer MFLOPS. Business interests may prefer Throughput, since it measures performance from an input and output standpoint.

In TABLE 1 is the evolution of the Intel CPU showing: the model, the year of introduction, MIPS, internal data path and clock rate measured in megahertz. Other CPU examples could be used, but Intel has always been an industry leader and was chosen for this example. [Ref. 4:p.S/8] [Ref. 5:p. 42] [Ref. 6:p. 110]

TABLE 1 **CPU EVOLUTION**

CPU	**Introduced**	**MIPS**	**Data Path**	**Clock (MHZ)**
4004	1971	000.06	4	1
8080	1974	000.29	8	2
8086	1978	000.75	8	4
8088	1979	000.75	16	4.77
80286	1982	001.5	16	16
80386	1985	005.0	32	33
80486	1989	027.0	64	66
Pentium	1993	100.0	64	66

Other models of the basic Intel CPU are available. For example, the 80386 is available in a 80386SX model has a 16 bit external bus to communicate with peripherals and a 32 bit internal bus. The 80386DX and the 80486SX are 32 bit internal and external. The 80486DX has the 80487 numeric coprocessor installed with the CPU. All models shown in TABLE 1 are top of the line in architecture and clock speed. Faster clock speeds are more expensive; slower clock speeds are available.

CPU's have yet another classification, they may be Complex Instruction Set Computers (CISC) or Reduced Instruction Set Computers (RISC). CISC are the standard CPU's found in most PC's. RISC is accomplished by reducing the instruction set, increasing the number of CPU registers and using instruction pipe lining. [Ref. 7:p. 434] The very fast and powerful RISC is desirable for high end operating systems like UNIX and WindowsNT.

C. MEMORY

CPU's operate with memory in a hierarchical fashion. [Ref. 2:p. 235] The actual portion of memory that interacts with the CPU is the register. When the computer is turned on the first memory read into the registers comes from Read Only Memory (ROM). This allows the computer to perform some self tests and boot up for operation. The disk drives will be checked for a system disk and if none are present a system start up routine may be read from a fixed disk. For example, if MS DOS is installed on the fixed disk and the floppy disk drives are empty, DOS will take control of the computer using the fixed disk.

Some computers may be equipped with a CD-ROM in addition to the fixed disk. CD-ROM has a large storage capacity usually in excess of 600 megabytes. Here is where the hierarchy may begin to form a triad of three factors: cost, size, and speed. The rule is simple. The closer the memory is to the CPU, the smaller its size, the higher its cost, and the greater its speed.

The size of a given register is normally going to match the size of the bus. Most registers will hold 32 bits or four bytes [Ref. 2:p. 124]. The registers receive memory addresses, data, and instructions from main memory. The size of the main memory may range from under one megabyte to sixteen megabytes or more. Programs are normally read into main memory from an internal fixed disk. Common sizes for fixed disks range from 20 megabytes to 200 megabytes or more [Ref. 1:p. 154] [Ref. 2:p. 209]. The larger operating systems like OS/2 ,UNIX and WindowsNT are now available on CD-ROM. CD-ROM can hold the equivalent of 600 high density floppy disks or 683 megabytes [Ref. 8:p: 13]. Register to CD-ROM is a hierarchy of small to large, bytes to megabytes.

The cost of memory is extremely volatile so specifics are avoided. Normally a user will never expand the number of registers because they are an integral part of the CPU. (Unless the mother board is replaced.) Occasionally the user will expand RAM or what is sometimes called main memory to improve system performance. More often the user will buy a new fixed disk so the computer will hold more programs internally. The trend today is for the user to buy a CD-ROM that houses vast amounts of information. This forms a cost hierarchy that ranges from most expensive registers to the least expensive CD-ROM. (The cost considered here is per unit of memory and not an overall cost of a given unit.)

Recall the fastest Intel CPU operated at 66 MHZ. Assuming the register is receiving new information every other clock cycle means the information rate for the register is actually 33 MHZ. Another way to view how fast the information is going into the register is in a time domain rather than a frequency domain knowing that time and frequency are reciprocals of one another. The 33 megahertz rate translates into new information in the register every 30 nanoseconds (10^{-9} seconds).

The main memory interacting with the register is accessed every 80 to 100 nanoseconds (ns). The fixed Disk can be accessed every ten to 100 microseconds (10^{-6}). The CD-ROM can be accessed every 300 milliseconds (10^{-3}) (ms). To resolve these differences in speed a buffer area called a cache may be placed between any two of these memory hierarchies. The information must pass through this hierarchy to be processed by the CPU. These time delays give the CPU time to do other things while waiting for data. TABLE 2 summarizes the memory hierarchy by memory component, component size and access time. Units of nanoseconds and milliseconds are used for easier comparison.

TABLE 2 MEMORY HIERARCHY

Component	Size	Time
Register	4 bytes	30 ns
Main Memory	1 to 16 MB	80 to 100 ns
Fixed Disk	20 to 200 MB	0.01 to 0.1 ms
CD-ROM	683 MB	300 ms

Other issues must be taken into account for memory usage. The memory must be addressable. The number of address lines on the address bus raised to that power of two gives the addressable space. Other tricks like virtual addressing and paging information in and out of memory is beyond the scope of this discussion. All modern operating systems accomplish these tasks in various ways.

D. NETWORK COMMUNICATION

Networking has become an integral part of computer operations. Many users are familiar with the modem, but fewer users may be knowledgeable about the network interface card (NIC) and terminal emulation. Modems and NIC's are hardware devices. This discussion is not an exhaustive presentation of networking or networking devices, but rather a brief overview of local area networking. A very generic look at modems and NIC's followed by two common terminal emulations is presented. (The issues concerning protocols are delayed until specific operating systems are discussed.)

Modems, whether internal or external to the computer, provide a quick and easy way to achieve network access with a stand alone system. America Online, CompuServe, and Prodigy are some of the commercial vendors that provide services via modem. A modem takes the computer's digital format and translates it into analog to transmit the signal over an analog phone line. [Ref. 9:p. 69] The modem on the other end reverses the procedure. The name modem is deri-ed from the fact that the signals are modulated and later demodulated.

Moving from a stand alone modem to a local area network (LAN) introduces the need for NIC (or LAN adapter). [Ref. 10:p. 17] Other hardware requirements like cabling and connectors are also needed, but the emphasis here is on the hardware native to the user computer on the LAN. The NIC is connected to the cabling giving access to the LAN. The LAN may be peer to peer where all computers have equal status or the LAN may be client-server. In one mode of client-server the server provides file server and print server services for the clients. The server will also have a NIC. Both client and server will have a network operating system (NOS) installed.

The NIC has two basic parts. [Ref. 9:p. 272] The bus interface unit (BIU) connects to the computer's input/output bus. The communications interface unit (CIU) connects to the medium or cabling system of the LAN. The NIC has a transmitter/receiver called a transceiver that transmits and receives data from LAN cabling. The NIC is also responsible for media access control (MAC), meaning that it must know when the computer can transmit.

The two types of MAC are contention and managed. [Ref. 10:p. 88-91] The two methods used to accomplish MAC are CSMA/CD and Token Ring respectively. CSMA/CD stands for carrier sensing media access with collision detection. In this contention environment, the NIC will listen for a carrier signal and if all is clear will transmit. If a collision is detected from another station the transmitting NIC will wait a variable length of time and transmit the information again. In Token Ring the transmitting computer must have the token to transmit. The token is passed around in circular or ring manner. Heavy LAN usage tends to make Token Ring more efficient since collisions are avoided.

From a PC on a LAN equipped with a modem a user may access a mainframe computer such as Amdahl or IBM. When this is done a special piece of software is needed to emulate a mainframe terminal. Common software for emulation is SimPC and 3270 Emulation. They allow the PC to emulate the 3270 terminal on the mainframe [Ref. 9:p. 380]. The keyboard on the PC is mapped to the terminal so the PC keys have a new meaning.

Emulation may be supported by modem or through a coaxial connection between a LAN server and a mainframe. Another popular piece of emulation software is SoftPC. SoftPC allows Macintosh users to emulate a PC. Emulation is important to expand user hardware and software options.

One of the main issues concerning networking is the use of protocols. Protocols for LAN's and wide area networks (WAN's) are covered with the applicable operation system that supports them.

E. OPERATING SYSTEMS

As mentioned in the introduction, the operating system provides the user interface to the computer and an interface for user programs to the hardware. These are only part of the operating system functions. Other functions include managing program processes, files, and devices. A generic look at these functions followed by a brief mention of application programs, compilers, and interpreters is given here.

The operating system brings program processes or tasks to the CPU. A process has three possible states: running (executing on the CPU), ready (waiting to run), and blocked (unable to run until some other event happens) [Ref. 7:p. 52] [Ref. 11:p. 30]. Process and task are used here interchangeably. When the CPU is able to run more than one process it is called "multitasking". (Multiprocessing means more than one processor or CPU.) The CPU has time to run more than one process due to the time lags in the memory hierarchy.

Not all operating systems support multitasking; the ones that do can support it in one or more ways. *Preemptive* multitasking is considered true multitasking. Priorities are calculated for the active processes and the highest priority process gains control of the CPU. The remaining two types of multitasking are not considered true multitasking by operating system purists.

Cooperative multitasking relies on special, "well behaved", programs designed to give up CPU time cooperatively without operating system intervention. *Time slice* multitasking allows user intervention to assign resources to a process. These resources are a percentage of CPU time and main memory (RAM). The user can set a type of high priority by assigning a large percentage of CPU time and a large amount of available RAM.

Authors vary on strict definitions of multitasking [Ref. 10:p. 132,133,171,211,261]. The definitions for multitasking used here are:

* Preemptive - the operating system, without user intervention, calculates a process priority and assigns CPU time and RAM. (Examples are UNIX and OS/2.)

* Cooperative - programs are written to keep track of the CPU time used and give up CPU time at regular intervals. This is a function of the application program and not the operating system. (Examples are Microsoft Windows 3.1 applications and Apple System 7 Savvy.)

* Time slice - the user can assign a percentage of CPU time and a portion of available RAM. (Windows 3.1 and System 7 both allow this function.)

Taking multitasking one step further is the idea of *multithread* processing. [Ref. 7:p. 587] [Ref. 10: p. 212] Each executable path through a process is a thread. By keeping the threads active a process can run threads in parallel for even faster execution. Multithread processing requires special coding.

In a single CPU environment multitasking is an illusion. The CPU is switched between tasks or processes so fast the user has the appearance of many things happening at once. When a process is blocked it is kicked out of the CPU and another process is brought in for execution. The most common reason for a block in a process is waiting for some input or output to a file (disk drive etc.) or other device.

File systems differ between operating systems. Some operating systems support more than one file system. For example, OS/2 supports the MS DOS file system that differs from its own. File systems are always closely joined to input and output. Handling for input and output to a specific device is rather uniform in the way an operating system is implemented. The operating system is kept as generalized as possible where devices are concerned. The operating system will not command the device directly, but will use a "device driver" instead. Device drivers may be hardware or software. The device driver is device specific and is usually provided by a device manufacturer or software vendor. [Ref. 11:p. 206] The device driver interfaces the hardware or software to a particular operating system.

A similar procedure is used to interface user application programs to an operating system. Application programs include: drawing programs, spreadsheets, word processors, and others. Application programs are written to a specific operating system. The operating system designers provide documentation on Application Program Interfaces (API's) to application program developers. These API's consist of operating system function calls where the application program can use the functions of the operating system.

For users who may want to write their own applications, operating systems provide an environment for compilers and interpreters. [Ref. 1:p. 206,208] A compiler, such as the UNIX C++, translates source code written in C++ language rules (called syntax) into machine binary code. This conversion is accomplished in one pass through the code. The computer can then execute the compiled binary code. Other programs called linkers and binders assist the compiler in memory addressing.

An interpreter, such as DOS BASIC, recognizes commands, translates the command and executes before going to the next command. The interpreter must translate and execute with each pass. This is much less sophisticated than a compiler.

Clearly, the *operating system* affects every aspect of computing from hardware to application software. This most important piece of software justifiably deserves a close study prior to selection.

II. UNIX

A. UNIX BACKGROUND

To understand the UNIX operating system a brief history and current status of UNIX is in order. Using a PDP-7 computer and the predecessor of the programming language C called B, Ken Thompson developed an early version of UNIX (Uniplexed Information and Computing Service) [Ref. 11:p. 267]. [Ref. 7:p. 572] [Ref. 10:p. 207] Thompson was later joined by Dennis Ritchie and they rewrote the UNIX operating system in C. This version of UNIX written in C originally ran on the PDP-11 series computers. Current versions of the UNIX operating system are written in C++.

Ritchie and Thompson published a paper on UNIX in 1974 and ten years later received the ACM Turing Award for computing excellence. [Ref. 11:p. 267] The paper stimulated the interest of many universities which prompted requests to Bell Labs for a copy of the UNIX operating system source code. At that time the parent company of Bell Labs, AT&T, was a regulated monopoly. AT&T could not compete in the computer industry and was willing to license copies of the UNIX operating system for a nominal fee.

Fortunately, most universities had the PDP-11 series computers in their computer science departments. The operating systems installed on most university computers were woefully inadequate. Furnished with the UNIX source code, professors and students enhanced the UNIX operating system by finding and fixing bugs and making improvements. Symposiums held on the UNIX theme led to the first de facto academic UNIX standard called Version 6. The USENIX users group was established to support this standard in 1975 [Ref. 7:p. 572]. The publication which supported Version 6 was the UNIX Programmer's Guide Sixth Edition [Ref. 11:p.267]. Version 6 was soon replaced by Version 7 which was the last time the UNIX operating system was covered under a single standard.

The goals of UNIX have remained relatively stable through the years. UNIX is an interactive time sharing system, designed by programmers for programmers. The design is multitasking for multiple users. Multitasking allows the CPU to run several parent and child processes, at times in parallel. Multiple users can share information on a restrictive basis. UNIX supports three user domains: the programmer, the programmer's group and others. Functions supported among users are read, write, and execute.

In the 1980's major events took place to shape the future of UNIX. [Ref. 7:p. 573] [Ref. 10:p. 207] Microsoft's commercial version of UNIX hit the market. The operating system name was XENIX. XENIX was designed to run on 16-bit microprocessors. To enhance the commercial UNIX product Microsoft Corporation added hardware error recovery, shared data segments and improved interprocess communication. During the same year (1980), the University of California at Berkeley, with grants from DARPA (Defense Advanced Research Projects Agency), now ARPA, had developed and released their own version of UNIX known as Berkeley UNIX or BSD (Berkeley Software Distribution). BSD supported distributed computing (peer to peer) on Digital Equipment Corporation (DEC) VAX mid-size computers.

With the divestiture of AT&T in 1982, the company was free to compete in the computer industry. [Ref. 7:p. 573] [Ref. 11:p. 268] Reacting quickly AT&T released their own "official" version of the UNIX operating system. AT&T's first serious UNIX operating system marketing attempt began with System III. System III offered remote job entry, source code control system and system accounting routines. System III evolved into System V which offers other system enhancements. Unfortunately, the AT&T System V and Berkeley UNIX are not compatible. Current releases of each operating system are System V Release 4 (SVR4) and BSD 4.4.

Berkeley introduced Wide Area Networking (WAN) to UNIX with the inclusion of TCP/IP (Transmission Control Protocol/Internet Protocol). [Ref. 10:p. 231] [Ref. 11:p. 269] The TCP/IP network protocol has become a de facto standard in the United States and is a DOD military standard. TCP/IP is in far greater use than the official standard supported by the International Organization for Standardization. Berkeley also added several utilities such as the "vi" editor and the "csh" shell along with Pascal and Lisp compilers.

These improvements prompted companies such as Sun Microsystems to originally model their SunOS after Berkeley's BSD instead of AT&T System V, but currently SunSoft follows SVR4 AT&T standards for both its Solaris and INTERACTIVE UNIX operating systems [Ref. 12:p. 2]. The vast majority of all UNIX operating systems support TCP/IP.

Many other competitors have entered the UNIX market with versions of the UNIX operating systems. Apple's version called A/UX was originally released for the Macintosh II in 1984 [Ref. 7:p. 772]. IBM's version called AIX is supported by the Open Software Foundation (OSF) that was established in 1988 [Ref. 7:p. 605]. The lack of a comprehensive UNIX standard (see POSIX) has limited the sale of UNIX in the system software market.

Milestones in the history if UNIX include:

- 1970 UNIX name coined
- 1973 UNIX rewrote in C
- 1974 Ritchie & Thomas UNIX paper published
- 1975 UNIX used at university level on PDP-11
- 1979 UNIX Time-Sharing, Seventh Edition
- 1980 Microsoft released XENIX
- 1980 Funding for Berkeley UNIX
- 1982 AT&T released System III with RJE
- 1984 ACM Turing Award to Ritchie & Thomas
- 1986 AT&T System V Interface Definition (SVID)
- 1987 Apple releases AU/X for Macintosh II
- 1988 AT&T System V Release 4 (SVR4)
- 1988 OSF founded to support IBM's AIX
- 1993 Berkeley withdraws from BSD

B. UNIX and POSIX

The Government has designated UNIX as the operating system standard for its departments and agencies. [Ref. 11:p. 270] [Ref. 13:p. 8-1] UNIX has several implementations in the market place that are not portable across hardware platforms. In an effort to make UNIX portable across platforms the National Institute of Standards and Technology (NIST) under the Department of Commerce has adopted the *Portable Operating System Interface for Computer Environments* Standard, (acronym POSIX). The "ix" portion of POSIX indicates the operating system portability is for the UNIX system [Ref. 11:p. 269].

Various attempts to *completely* standardize UNIX have failed. [Ref. 11:p. 269] [Ref. 13:p. 8-2] Two distinct incompatible versions of UNIX emerged in the late 1980's (System V and BSD). Vendors adding nonstandard enhancements further complicated UNIX compatibility. AT&T, Berkeley and consortiums of various vendors have tried to standardize UNIX. One of the initial attempts at standardization by AT&T was SVID (System V Interface Definition). SVID defined file formats and system calls in an attempt to standardize System V. SVID was ignored by the Berkeley UNIX group. The nonstandard binary program formats limited the commercial success of UNIX because software vendors could not write package UNIX programs with portability across UNIX systems.

Finally, a neutral body was brought in to reconcile System V and BSD. [Ref. 7:p. 603] The body chosen was IEEE (Institute of Electrical and Electronics Engineers). IEEE began the POSIX project with the goal of standardizing the UNIX operating system in an effort to achieve portability across UNIX platforms. The IEEE designator for POSIX is 1003.x (where "x" represents a series number dealing with a specific function or service). The designator IEEE 1003.0 may also be referred to as POSIX.0, which in this case refers to a "Guide and Overview" of the POSIX standards.

The fundamental idea behind POSIX is that a software vendor that supplies a program which is POSIX compliant has used only procedures defined by the POSIX standards. When POSIX is fully enacted this will ensure the program will run on a conforming UNIX system. POSIX can best be described as an incomplete standard. [Ref. 13:p 8-2] The problem is POSIX *does not* currently address all of the functions needed to implement operating system software. The reason POSIX is incomplete is due to the way IEEE decided to implement the standard.

IEEE took the intersection of features found in System V and BSD rather than the union of features. [Ref. 7:p. 13] [Ref. 11:p. 270] In other words, if a feature is present in both System V and BSD the feature is included in the IEEE standard otherwise it is not addressed. Examples of services not addressed by IEEE standards are: login services standards, checkpoint and restart standards and resource limit standards, just to mention a few [Ref. 13:p. 8-28,8-32,8-34].

The result of the IEEE intersection of System V and BSD creates a standard that closely resembles the academic UNIX prior to the AT&T divestiture, namely Version 7 [Ref. 11:p. 270]. It is possible for vendors to be POSIX compliant and still conflict between themselves. [Ref. 14:p. 594] A group of vendors responding to AT&T control of a large part of the UNIX market set up a consortium called OSF/1 (Open Software Foundation). The purpose of OSF is to produce a system that meets IEEE standards. The additional features of OSF/1 such as X11 Windows and MOTIF graphical user interface makes the OSF/1 system incompatible with System V and BSD. The AT&T consortium response to OSF is UNIX International (UI).

The requirement of functions outside the realm of POSIX has lead to "Extensions" to POSIX. It is the extension portion of the operating system which leads to incompatibility and non-portability between vendors. Different implementations that conform to POSIX (e.g. BSD, OSF/1, UI, and other vendor products) often support the same function differently. The names of UNIX system calls may be identical between vendors. Vendor incompatibilities arise from differences in data types of the function, data types of the arguments, the return values, the header files and the symbolic error values [Ref. 13:p. 8-2]. These systems continue to evolve in different directions.

Three sets of nearly identical documents govern the POSIX "Application Program Interface" standard. [Ref. 13:p. 8-1] [Ref. 15:p. 1] The IEEE 1003.1, POSIX *Interface* for Computer Environments was originally adopted by NIST in 1988. NIST adopted IEEE 1003.1 calling it FIPS 151-1 (Federal Information Processing Standard). The most current edition is FIPS 151-2 which was released in May 1993. The goal of FIPS is to give the federal government more effective control over information resources via compatibility and portability. NIST currently conducts POSIX compliance tests and publishes a register of POSIX compliant UNIX products. Mirroring the IEEE 1003 series standards are the ISO/IEC 9945 series standards. The ISO/IEC 9945-1 standards are exactly the same as IEEE 1003.1.

X/Open is the nonprofit, international UNIX standards organization [Ref. 14:p. 607]. It provides a common ground for two industry leaders. The UNIX International group backing System V Release 4 (SVR4) includes: AT&T, Data General Corporation, Sun Microsystems, and Unisys Corporation. [Ref. 14:p. 594] The Open Software Foundation supporters of OSF/1 consists of: IBM, Digital Equipment Corporation, Groupe Bull, Hewlett Packard, and Nixdorf Computer AG. Vendor specific features and proprietary barriers result in UNIX non-portability across platforms.

C. UNIX PROCESS MANAGEMENT

UNIX is a multiprogramming, multitasking system, so several independent processes from different programs appear to be running at the same time. The process is the simplest level of a program and can be viewed as an individually controllable computation entity or a task to be accomplished. Processes can be running (or executing), ready to run (waiting) or blocked from running pending some action. The idea of multitasking is to keep the microprocessor as busy as possible and not to wait for a process that is blocked.

UNIX primarily uses preemptive multitasking. As an example, at timed interrupts of ten times per second, the UNIX scheduler of the SunOS examines the processes that are ready to run. [Ref. 7:p. 582] The scheduler algorithm is a finite amount of definable steps to accomplish the scheduling task. Based on a priority adjustable algorithm the scheduler takes into account the following:

* process priority,
* amount of CPU time recently used, and
* amount of time the process has been on hold.

These determine which process the scheduler will start next. Process priority is recalculated once per second by the scheduler.

The scheduling algorithm is set to forget a portion of CPU time used by a particular process based on the number of competing ready processes waiting for the CPU. [Ref. 7:p. 583] [Ref. 10:p. 212] This means that the running process is not indefinitely penalized for past CPU usage and that it will receive more time with less competitors present. New processes have a higher priority if they have not received any CPU time. The result is three favorable factors for multitasking:

* Processes that require little CPU time, (called I/O bound processes), get favored treatment.
* Processes that take more CPU time, (called processor bound processes), are not postponed indefinitely because the scheduler is programmed to forget some of the CPU time used by a process.
* The system adjusts to the process environment based on the number of ready processes.

Programs that run on UNIX all start as a single process. [Ref. 10:p. 209] [Ref. 11:p. 281] [Ref. 14:p. 208] UNIX uses a *fork call* to divide a process into two or more processes if needed. The fork call makes an exact copy of the original process. The original process is the parent process and the copy is the child process. The child has the same priority as the parent since it is an exact copy, but each process is given its own memory space to run and is treated independently. To distinguish the parent from the child process the fork call assigns a process identifier (PID) to each process.

Interprocess communications can take place in a variety of ways in UNIX. [Ref. 7:p. 583] [Ref. 11:p. 283] If processes share a memory area for communications a semaphore data structure (usually binary) is used to lock the shared resource. Communications paths between processes may be permanent, which are called Named Pipes or they may be a First In First Out (FIFO) queue of bytes, simply called a Pipe. Similar to Named Pipes and Pipes, Sockets are two-way communications lines between processes that can be created and destroyed dynamically. Sockets are particularly useful because they support the use of protocols such as Berkeley's TCP/IP. (Berkeley introduced sockets while AT&T introduced Named Pipes and Pipes). AT&T System V currently uses Message Queues for primary interprocess communications.

Signals are actually software interrupts similar to hardware interrupts. [Ref. 7:p. 583] [Ref. 11:p. 283] A process can send a signal to another process and the receiving process can act upon the signal or not. Unlike hardware interrupts, signals have no priority system. Because signals have no priority system and are limited in scope of communication. They are not intended for interprocess communications. Signals can only be sent between the same process group which are parent, children and other direct descendent. UNIX supports approximately 20 different signals.

To keep track of all the processes generated, UNIX employs a process table for all running processes. [Ref. 10:p. 210] [Ref. 11:p. 300] Information maintained in the process table falls under four areas: scheduling, memory, signals, and miscellaneous information. Scheduling parameters include all the needed information for the scheduler. This includes process priority, amount of CPU time used, and how long the process has been waiting for CPU time. Memory information is simply address information (pointers) to where various program parts reside on disk or in main memory, such as data, stack and text segments. The program in main memory is the memory image. The process is the execution of the image [Ref. 14:p. 186].

Signal information stored for each process includes: which signals are being ignored (blocked), which signals are pending, and which signals have special handling routines. [Ref. 10:p. 210] The miscellaneous information category includes ownership and process relationship information. Ownership is information about user and group identification used for security. The process relationship information has the process identification numbers of the immediate family of a group processes. The current status of the process is also filed under miscellaneous.

Maintaining a process table for each process causes a great deal of processing overhead. [Ref. 7;p. 587] Each process has its own program counter, set of registers and address space. Some programming languages provide the ability for parts of a process to execute in parallel. Parallel processes have various threads of control which are discrete execution paths. Several threads of a process executing in parallel are called multithread execution. Multithread execution has two advantages. First, it allows the operating system to take full advantage of multitasking capabilities and second, there is a reduction in process overhead.

Unlike processes ,threads share address space so there is less protection between threads. [Ref. 7:p. 587] [Ref. 10:p. 212] [Ref. 11:p. 508] This lack of protection is not a problem since threads belong to the same process. Sun Microsystems version of the UNIX operating system (SunOS), calls this multithread execution with low overhead *lightweight processes*.

The high efficiency which the UNIX operating system controls the CPU enables multiple programs to run in what appears to be a concurrent manner. Programs are said to be running in the foreground or the background. [Ref. 14:p. 189,190] Each program is composed of processes. The processes running in the background are called *daemons* [Ref. 11:p. 279] [Ref. 14:p. 197]. Typical uses for daemons are: electronic mail services, handling a print queue, managing memory and timer services. Implementation of daemons in UNIX is elegant and simple since each daemon is a separate process that is independent of other processes.

The minimum CPU requirement for running the UNIX operating system is the Intel 80386 or the Motorola MC68030 [Ref. 10:p. 205]. Memory requirements vary between vendors. Typical UNIX PC's require 12 megabytes of RAM and 200 megabytes of fixed disk space [Ref. 12:p. 12].

D. UNIX FILE SYSTEM and FILE SECURITY

A UNIX file is a collection of addressable bytes containing arbitrary information that the user has chosen to place into it [Ref. 7:p. 577]. UNIX makes no distinction between ASCII files and binary files, so the meaning of the file is left to the owner [Ref. 11:p. 276]. The UNIX file system is hierarchical system with the root node at the origin. [Ref. 7:p. 576] [Ref. 10:p. 216] Security is provided to all files via a nine bit protection called *rights bits* [Ref. 11:p. 277].

File extensions that are added to the end of a file name such as .EXE or .SYS for executable and system files, respectively, are not enforced in UNIX. [Ref. 10:p. 218] File names are normally up to 14 characters, but can be up to 255 characters long in Berkeley UNIX. The result is that very flexible and descriptive file names may be used.

The hierarchical file system supports directories and subdirectories. [Ref. 10:p. 216] [Ref. 11:p. 287] Unlike DOS and OS/2 that use the back slash (\) to delineate the path name to a file, UNIX uses a forward slash (/). Files can be accessed by using either the *absolute path name* or the *relative path name.* Starting at the root directory and specifying all subdirectories down to the file name is the absolute path name. Starting at the current or working directory and specifying only subdirectories below the current directory down to the file name is the relative path name.

The file system always knows the user's *current* or *working* directory. Therefore, the user can access the desired file that is under the current directory without referencing through the root directory because the file name used is assumed to be under the current directory. The mount system call can be used to attach a file system to a directory in another file system [Ref. 7:p. 577]. Unmount reverses the procedure.

The disks that employ the UNIX file system have a defined data structure resident on those disks. [Ref. 7:p. 577] [Ref. 10:p. 217] [Ref. 11:p. 307] The physical structure can be viewed as a series of contiguous blocks up to some point and then the address (or pointer) of the next set of contiguous blocks. The first block (block 0), is not used by UNIX and is usually reserved for computer boot code. Block 1 is called the *super block*. The super block contains information that defines the file system. Destruction of the super block renders the system unreadable because a list of actual data file blocks and a list of free blocks are maintained here. All block allocations are in fixed-sized blocks.

In addition to lists, the super block also contains an array of *inodes*. [Ref. 7:p. 578] [Ref. 10:p. 219] [Ref. 11:p. 308] An inode is sixty four bytes long and contains information about a particular file. There is an inode for every file and directory on the disk. UNIX also uses directories to relate the name of a file to its inode number. These directory lists are separate from the file description on disk so more than one directory can point to the same file on disk. UNIX calls this capability *linking*.

The concept of linking is important because a non-directory file may appear in many different directories under different names. [Ref. 7:p. 577] [Ref. 10:p. 217] [Ref. 11:p. 288] The number of names a physical file has is called the link count. Directory entries for the files are called *links*. Files can exist independently of the directory entries. Links pcint to inodes that point to the physical file on disk.

An inode contains the information on: the number of links to a file, time modifications, file type, physical size, location, ownership and *permissions* [Ref. 7:p. 578]. Unix *permissions* fall into three categories: the owner, the owner's group and all others. [Ref. 7:p. 578] [Ref. 10:p. 215] Each category has three options: read, write and execute the file. The owner may also elect to grant no permissions indicated by a dash (-) rather than r,w,x used for the read, write and execute permissions. Nine bits are used in this file protection scheme, three for the user, three for the user's group and three for all others. Only the user and the system administrator can modify the protection of a file. The system administrators are also called the superusers because they alone have control over system files.

UNIX time modifications are similar to other file systems. Time modifications include: the time the file was created, last used and modified. The time the inode was last modified is also maintained.

File types are: ordinary, directory and special. The first two are the standard files that most users are familiar with seeing. [Ref. 7:p. 578] [Ref. 11:p. 291] The special files are used with input/output devices and are either character special files for eight bit serial transfers (one byte) or block special files for larger transfers typically. These are 512 or 1000 bytes for random access.

The physical size of the file will determine how the UNIX stores it. [Ref. 7:p. 578] [Ref. 11:p. 165] The inode points directly to the first ten blocks of a file. For small files this is usually all the memory space needed. As the file grows, UNIX employs an indirect addressing method that can extend three levels deep. Inode fields zero through nine are direct block addresses, field ten is a single indirect block address, field 11 is a double indirect block address and field 12 is a triple indirect block address.

For files larger than ten blocks of memory, field 10 of the inode contains the address of the single indirect block. This block contains the disk addresses of more disk blocks.

For example, if the block is 1000 bytes (1 K) and each address was four bytes, the single indirect block could hold up to 256 addresses (or 256 K of memory) before the double indirect block is needed. Following the same example, the double indirect block would contain 256 single indirect blocks that would each hold addresses of 256 data blocks. Double indirect address will hold up to 65,536 addresses and beyond that the triple indirect blocks hold up to about 17 million addresses.

As the file grows the time necessary to access the entire file past the first ten blocks also grows because the levels of addressability increase. A large file may require several pointers to be used to retrieve the entire file due to the indirect addressing method used with UNIX.

No organization is enforced on the file. The organization is decided by the application program that created the file. This emphasizes the notion that UNIX treats files as addressable bytes defined by the user or application program.

To facilitate sharing files across a network two systems are dominate among UNIX users, NSF and RFS. [Ref. 7:p. 601,602] [Ref. 10:p. 231,729] Network File System (NFS) was developed by Sun Microsystems and released in 1984. The design of NFS was for wide area usage across several dissimilar networks. NSF is based on the Remote Procedure Call (RPC) and External Data Representation (XDR).

RPC enables programs to retrieve values from other computers on the network and XDR is the standard used to represent the data between different types of computers.

The Remote File System (RFS) was released in 1986 by AT&T to support distributed UNIX computing with System V Release 3.0. [Ref. 10:p. 727] RFS can operate using TCP/IP and other network protocols. AT&T's RFS and Sun's NSF are not compatible without an intermediate translation [Ref. 13:p. 8-10].

E. UNIX INPUT/OUTPUT SYSTEM

UNIX treats input and output (I/O) to devices as streams of bytes [Ref. 7:p. 590] [Ref. 14:p. 402]. Devices for I/O typically include terminals, disk drives, printers, and networks connected to the devices. This macro-level handling of I/O has the net effect of placing the burden of defining the streams of bytes on the application program. A common structure for UNIX operating system application program is a text stream of ASCII characters. The UNIX I/O system calls assume unstructured byte streams; no definition of the byte stream is assigned by UNIX. All structure for the byte stream is created by the application program.

To allow access to devices, UNIX integrated the devices into the file system as *special files*. [Ref. 7:p. 591] [Ref. 11:p. 290] Like other system files, special files are owned by the superuser (system administrator). By convention device special files are usually placed in a directory called "/dev". Each device is assigned a path name. For example, a line printer may be referenced by /dev/lp. The path name not only gains access to the particular device, but since the device driver is part of the file system it is afforded file protection. The disadvantage of embedding device drivers in the UNIX system is the requirement that they must be linked to the system by the system administrator.

Special files have two categories, block and character special files. [Ref. 7:p. 582] [Ref. 11:p. 290] Block special files are for random access and are used to access disks. The file consists of a sequence of numbered blocks. Each block is individually addressed and accessed. The goal of UNIX I/O system on block special files is to minimize the number of data transfers. UNIX systems place a buffer cache between the disk drives and the file system which minimizes the transfers needed.

A read cache is very easy to implement and is used even in the simplest operating systems. The operating system merely reads more data than what is requested thus reducing the number of disk accesses for sequential requests. Write cache is more difficult to implement and is handled differently between specific versions of UNIX.

Character special files are primarily used for devices that use character (byte) streams and where random access is not needed. [Ref. 7:p. 578] [Ref. 11:p. 291] Typical character special files include : keyboard, mouse, network and printer. Files may be "read only" for a keyboard or a mouse; they may be "write only" for a printer or they may be "read and write" for a network communication device.

File I/O may be redirected using greater than and less than signs to override standard inputs from keyboard and mouse and standard outputs to the screen. [Ref. 7:p. 574] [Ref.

11:p. 275] [Ref. 14:p. 48] UNIX automatically opens three files upon start up: standard error and standard output to the display and standard input for the mouse or keyboard. Redirection of I/O and opening of standard files is very similar to DOS.

The UNIX I/O system uses five primary system calls which are: open, close, read, write and long seek (lseek) [Ref. 7:p. 592]. The first four calls are straightforward functions, the last system call, lseek, is used for random access. System calls are only allowed with special files. Associated with each call is a unique file descriptor.

The file descriptor is an integer. It defines files and special files to a process. [Ref. 7:p. 591] [Ref. 11:p. 15] They let the process determine the status of a file or special file. Descriptors are essential to the implementation of named pipes and sockets, which are both special character files. DOS calls a file descriptor a file handle.

Recall that named pipes are static communication linkages between processes while sockets are dynamically created as needed for I/O across the network. Network I/O is an integral part of the UNIX operating system.

When using a Berkeley socket for network I/O, each socket supports a different type of networking and protocols between systems. [Ref. 7:p. 597] [Ref. 11:p. 292] Looking at the sockets generically, without regard to a particular protocol, three basic types of sockets are available. The three most common are:

* reliable connection oriented byte stream,
* reliable connection oriented packet stream,
* unreliable packet transmission.

The byte stream is commonly supported by the UNIX to UNIX Copy Programs (UUCP) using a modem. [Ref. 10:p. 230] [Ref. 14:p. 494] UUCP is a collection of programs that perform functions of: mail transfer between systems, remote command execution, and remote file transfer. Byte streams are controlled by UNIX directly; packet streams require additional software.

Pipes between processes may use a reliable connection byte stream or packet stream for communication [Ref. 11:p. 292]. The unreliable packet streams are supported via User Datagram Protocol (UDP) [Ref. 11:p. 293]. UDP represents best effort delivery without any guarantee.

Reliable conectionless packet streams in UNIX are supported by Transmission Control Protocol/Internet Protocol (TCP/IP). [Ref. 10:p. 94] [Ref. 11:p. 293] Both TCP/IP and UDP protocols originated with DOD's ARPANET, which is now Internet.

An important feature of the UNIX networking concept is the *non-blocking* I/O also introduced by Berkeley [Ref. 7:p. 593] [Ref. 11:p. 410]. A blocking I/O is synchronous and is checked at regular time intervals. This can slow the system. A non-blocking I/O is asynchronous so the wait is averted. The non-blocking I/O is useful when monitoring several communication lines and wanting to read the data as soon as it becomes available. Traditional UNIX systems used a synchronous interface to a process caller. The calling process was blocked until the request could be satisfied.

The idea behind a non-blocking I/O is that the process can request to be signalled asynchronously when I/O becomes possible using a non-blocking descriptor. If the read or write can not be accomplished an error code is returned. Berkeley UNIX employs a multiplexing feature that enables a process to determine when any of several descriptors are ready for I/O [Ref. 7:p. 593]. The net effect is faster communications.

UNIX employs very advanced features in all facets to handle inputs and outputs. It is important to emphasize that while UUCP is a standard part of the UNIX operating system, TCP/IP and UDP are additional software packages required for network I/O.

F. UNIX USER INTERFACE

UNIX employs two types of user interface: command line interface, which is called *shell,* and *graphical user interface* (GUI). Each type of interface has three variations. Before addressing specifics about these interfaces, a look at the layers of software in the UNIX operating system is in order to provide a basis that will also apply to other systems.

The *user interface* is the outer layer of software and is normally perceived as being the UNIX system. The user interface via the shell or GUI provides access to text editors such as "vi", compilers such as C, and other utilities. These are collectively referred to as *standard utility programs.* [Ref. 11:p. 273] Just below the user interface resides the *standard library interface.* Standard library functions include: open, close, read, write, and fork. These functions are automatically available to the user interface. The user interface along with standard library functions comprise the *user mode.*

Directly below the user mode is the *kernel mode.* [Ref. 11:p. 273] The kernel mode may be viewed as the heart of the UNIX operating system because it provides the interface to the hardware via the *system call interface.* This is where the actual process and memory management takes place along with the file system and I/O system management.

The UNIX operating system is written in C (or C++). The actual grassroots communication to the hardware is device specific assembly language called *device drivers* and is supplied by the manufacturer of the device. The complied C language communicates directly with the assembly language to control the CPU, disks, keyboards, etc. Each peripheral device must be linked to the kernel in UNIX [Ref. 14:p. 214].

The hardware specifics are totally transparent to the user with the exception of speed in execution of commands. The user will see either the shell prompt or the GUI. [Ref. 7:p. 573] [Ref. 14:p. 283,560] The three most popular shells all contain "*sh*" as part of their file names. AT&T's first popular shell, the *Bourne shell* 1976, has the file name "sh" and displays "$" on the command line. The enhanced version of the Bourne shell, the *Korn shell*, uses the same prompt and is also an AT&T product. The Korn shell (file name "ksh") incorporated some of the features found in the third type of shell namely, the *Berkeley C shell* (file name "csh"). This shell was named C shell because the syntax (language rules) used are based on the C programming language. The C shell uses "%" prompt on the command line unlike the others.

The command line of the shell, indicated by the prompt, is expecting the name of an executable file and one or more arguments to act as input to the file. The job of the shell is to interface with the standard library. [Ref. 7:p. 574]

[Ref. 11:p. 275] This ensures the fork call is made to create the child process which is created from the parent and to open three files. The three files are for standard input (usually the keyboard), standard output and standard error (usually the display). These are used for reads, writes and error messages respectively.

UNIX allows redirection of standard input and standard output. [Ref. 7:p. 574] [Ref. 11:p. 275] [Ref. 14:p. 68] Standard input is redirected by "<" and standard output is redirected by ">", in other words "less than is input redirection" and "greater than is output redirection". This is extremely useful when input or output is to a file. Additionally, the UNIX shell syntax allows for an output to be appended to a file rather than replacing it. In typical UNIX shorthand elegance this is accomplished by using ">>".

UNIX commands in shell can be placed together and executed in succession using a vertical bar "|" called the *pipe symbol*. [Ref. 10:p. 221] [Ref. 14:p. 23] The collection of commands is sometimes referred to as a *pipeline*. This allows the output of one command to be the input to the next command which UNIX calls a *filter*. A file containing shell commands is called a *shell script*. The user may have the script execute upon start up.

UNIX shell supports traditional programming language constructs (case, for, if-then-else, and while). [Ref. 7:p.

574] [Ref. 14:p. 190] With the power of UNIX processing the user can also designate a program to execute in the background, such as a compiler, using "&" while another program runs in the foreground, such as a word processor or spreadsheet.

The DOS command line is similar to features found in the UNIX shell. The DOS batch file is like a UNIX shell script. Redirection of I/O and pipe lining are the same and basic programming language functions are supported in both operating systems. These similarities may be carried one step further since the OS/2 operating system supports DOS and has provisions for UNIX.

Today most users tend to prefer the graphical user interface (GUI) over a command line as provided by shell. [Ref. 10:p. 226] [Ref. 14:p. 593] The X Window System, sometimes simply called "X", was first released by Massachusetts Institute of Technology in 1984. The source code for "X" is publicly available. Digital Equipment Corporation was the first to commercially market "X" and presently supports DECwindows; however, most application programs written for "X" use one of the two leading industry standards, OpenLook or Motif.

OpenLook is supported by AT&T (UNIX Systems Labs) and Sun Microsystems (SunSoft) under the UNIX International (UI) consortium. [Ref. 10:p. 227] Motif is an IBM product supported by Open Systems Foundation (OSF). Santa Cruz Operation's (SCO) Open Desktop is based on OSF Motif. The AT&T and IBM windowing systems are fundamentally incompatible; however, a joint venture between UNIX Systems Labs and Novell called UNIVEL has released software called UnixWare that supports both Motif and OpenLook on an Intel-based workstation.

Icons differ on different versions of UNIX windowing systems to represent data files and programs. Commonalities include similar program manager windows and pull down menus. With all UNIX GUI's, a program may be started by: double click on a executable program's icon, double click on a program name in the file system window, double click on a data file with an association to an executable program, or open a window and type the program name.

G. UNIX NETWORKING

UNIX provides an environment that is very natural for networking due to the support for multiple processes and multiple users sharing resources. Basic capabilities of UNIX networking include NFS and RFS for file sharing and UUCP and TCP/IP that support file sharing as well as remote program execution, file transfer, and e-mail [Ref. 10:p. 230]. UNIX has several other protocols available. UUCP and TCP/IP are two of the most common and the focus here.

During the development of UNIX the two PDP series computers that were used were separated by a flight of stairs. Developers disliked climbing the stairs. The result was the earliest releases of the UNIX operating system supported UUCP (UNIX to UNIX Copy Program) to support networking on a peer to peer basis.

UUCP has evolved to several programs and related files that allow the user to carry out three primary tasks: remote file transfer, remote command execution, and mail transfer. UUCP is now classified as a point to point protocol for use with modems. Data speeds are typically 9,600 bits per second, but speeds may be higher [Ref. 10:p. 724] [Ref. 14:p. 492]. Data rates are transparent to the user because UUCP operates as a background process (daemon) and carries out transfers at the convenience of the operating system. The transfer will take place the next time the remote system is connected with the local system if a current connection is not already in progress.

The UUCP mail will notify the file recipient that a file transfer has taken place upon the next logon if the remote station is not currently connected. [Ref. 10:p. 734] [Ref. 14:p. 494] The recipient of the files can view the contents using a standard "uupick" command after a directory for placement is specified. UUCP support for remote execution is via the "uux" command. If the permission is granted by the remote system the results of the remote execution are displayed at the local system. The UUPC command will connect a properly configured MS DOS PC to the UUCP network [Ref. 10:p. 230].

For higher level networking across several different networks many UNIX systems support Transmission Control Protocol /Internet Protocol (TCP/IP). TCP/IP requires additional software to be ran on top of the operating system. TCP/IP is discussed here because U.C. Berkeley incorporated it in UNIX. All other operating systems such as MS-DOS, OS/2, and System 7 have add-on software to support TCP/IP so it will not be discussed in their respective sections.

Internet Protocol provides the basis for other protocols. IP is not designed as an end user protocol [Ref. 11:p. 434]. It forms the basis for TCP and UDP. Transmission Control Protocol (TCP) provides a virtual circuit between remote sites for reliable transmission with error detection and correction. [Ref. 10:p. 232] User Datagram Protocol (UDP) is connectionless "datagram" service which is a best effort for short message transmission without a guarantee of success. TCP/IP provides four basic services across a network. [Ref. 10:p. 232,721] They are:

* *File Transfer Protocol* (FTP) allows users to log on to remote sites to view directories and to send and receive files.

* *Simple Mail Transfer Protocol* (SMTP) uses store and forward techniques to serve as the basis for electronic mail.

* *Simple Network Management Protocol* (SNMP) enables several networks to be administered (managed) from a central location.

* *Telnet* provides a remote log on capability with a remote terminal emulation if necessary.

UNIX is a very flexible networking platform. TCP/IP and UUCP provide two separate means of file transfer, mail, and remcte execution. TCP/IP can be used with NFS and RFS. UNIX can also be used in the traditional client server LAN environment in addition to the peer to peer distributed environment.

E. UNIX ADVANTAGES and DISADVANTAGES

Some of the very same characteristics which may be considered advantages in UNIX may also be considered disadvantages. UNIX runs on the widest variety of hardware platforms of any operating system [Ref. 10:p. 234]. UNIX can be adapted to almost any hardware. Personal computers, workstations, and mainframes run UNIX systems such as XENIX, SunOS, and System V respectively. This wide range in UNIX varieties leads to the greatest downfall of the UNIX system. UNIX is incompatible between companies and is not portable across hardware platforms.

Despite the POSIX standards every version of UNIX requires a special version of an application program to run on that particular version. UNIX application programs are often sold by "value added" vendors who have to customize the application to the version of UNIX in use [Ref. 10:p. 237]. The incompatibility issues of UNIX translates directly into increased cost.

Additional cost is not just limited to application programs. The UNIX operating system is much more costly than DOS or OS/2 when implemented on a 80386 or 80486 computer. DOS does not measure up to UNIX functionality, but OS/2 has many of the desirable UNIX multitasking features.

The size and complexity of most UNIX systems usually requires additional memory, and a tape drive or similar device is needed for installation and backup. Unix may not be the best choice for small Local Area Networks (LANs) or for LAN's that cannot support a system administrator position. The system administrator must link new devices to the UNIX operating system, unlike most other systems where device drivers may be installed by the user.

UNIX is a good choice when multitasking computing power is needed for parallel processing, using multiple processors, or simulated multitasking is needed on a single processor. UNIX supports true multitasking on 16 or 32 bit CPU's. UNIX supports RISC technology on parallel and single processors. Sun Microsystems (SunSoft) produces INTERACTIVE UNIX and Solaris operating systems to meet these needs [Ref. 12:p. 1]. (INTERACTIVE and Solaris also have some DOS and Windows 3.1 compatibility [Ref. 12:p. 22].

UNIX uses preemptive multitasking in a multiuser environment with built in security. Log on is the first line of defence followed by file security as a second line of defence for unauthorized users. DOS and OS/2 provide none of these features without a LAN server or additional third party software. OS/2 provides preemptive multitasking, but for a single user. System 7 provides security but none of the other features.

The only operating system that is comparable to UNIX in total functionality is Microsoft's WindowsNT. An in depth discussion of UNIX and Microsoft WindowsNT is covered here because WindowsNT is not considered a PC operating system. Microsoft's selling points for NT are: that it is a single company product with a single common user interface and the network protocols integrated into the system. UNIX has many vendors. UNIX protocols are purchased separately.

UNIX should be used where the need exists, such as a multimedia multiuser environment, where security and speed are factors and system cost is justified. The payoff for a large complex operating system is the performance that may be achieved.

III. MS DOS

A. MS DOS BACKGROUND

DOS first appeared in the market place in August 1981. [Ref. 7:p. 633] [Ref. 10:p. 129] [Ref. 11:p. 317] The success of Microsoft DOS can be traced to two early developmental steps. First, IBM manager Philip Estridge was able to build an inexpensive sixteen bit personal computer from commercial off the shelf (COTS) parts from Intel, and second, Microsoft was able to win the race over Digital Research to the market place for a sixteen bit operating system. The 13 year history of DOS has been well documented by numerous sources so a detailed account may be given here.

Digital Research had dominated the market in operating systems for eight bit microprocessors (Intel 8080). [Ref. 7:p. 633] The operating system was called "Control Program for Microcomputers" or CP/M. Released in 1975, CP/M consisted of three major subsystems:

* Control Command Processor (CCP)
* Basic Input/Output System (BIOS)
* Basic Disk Operating System (BDOS).

Microsoft's first release of DOS resembles the CP/M operating system. CP/M originally had a great influence on DOS. Early versions of DOS had essentially the same three major subsystems functions.

When IBM originally asked Bill Gates, founder of Microsoft, to develop an operating system for the IBM Personal Computer (PC), Gates suggested Digital Research's CP/M. [Ref. 7:p. 635] [Ref. 11:p. 316] IBM's confidence in Microsoft came from the company's successful development of XENIX (a 16 bit version of UNIX), BASIC (a command interpreter for the Altair PC) and a file system using a File Allocation Table or FAT (which later became the basis for DOS). IBM went to Digital Research to inquire about CP/M-86 the new 16 bit operating system. When they found CP/M-86 well behind schedule IBM returned again to Microsoft and requested an operating system like CP/M.

Bill Gates was aware of a nearby company named Seattle Computer Products that produced memory boards for the 8086-based (16 bit) products. [Ref. 7:p. 634] [Ref. 11:p. 316] Gates hired Tim Paterson, who in 1979 had developed software called 86-DOS to test the sixteen bit memory boards. The DOS project began in April 1981 and was completed on schedule in August of the same year. This Intel-based operating system had two similar versions for the PC: MS DOS by Microsoft and PC DOS by IBM.

Early decisions on hardware drove software decisions and the accessibility of the PC market. Two important decisions by IBM helped shape the future of DOS. First, IBM decided to make the PC an *open system* [Ref. 11:p. 317]. This means the complete design described in great detail was available from PC vendors. Open system architecture nurtured the development of PC clones and gave Microsoft a software market outside of IBM. (Although functionally about the same, there are some subtle differences between PC DOS and MS-DOS.) Second, IBM's selection of the 8088 over the 8086 made the product less expensive. The 8088 handles internal transfers of sixteen bits each clock cycle and external transfers to peripherals eight bits at a time [Ref. 7:p. 634]. This did not pose any problems because most peripherals of the time communicated eight bits at a time. Selection of the 8086, which communicates at sixteen bits externally and internally, would have made the PC's less affordable.

The 8086 and 8088 are both equipped to handle a one megabyte address space. This is because twenty address lines are available to these two CPU's (2^{20} is approximately equal to one million). Early in development IBM decided to reserve 384 kilobytes for hardware uses leaving 640 kilobytes for DOS and DOS application programs. [Ref. 10:p. 163] [Ref. 11:p. 317] This lead to the 640K limit legacy DOS has today. The 640K limit has continued to plague DOS because new programs written needed to be compatible with 8088 architecture. Innovations to overcome this problem are discussed with the appropriate version of DOS that introduced a solution.

MS-DOS Version 1.0 consisted of 4000 lines of assembly language code that occupied twelve kilobytes of memory. [Ref. 7:p. 634] [Ref. 10:p. 129] [Ref. 11:p. 317] The operating system was organized into three files (similar to CP/M) which were:

* IBMBIO.COM for disk and character I/O system.
* IBMDOS.COM for disk and file management.
* COMMAND.COM for shell command processing.

Version 1.0 was compatible with CP/M which was important in the marketplace in order to gain greater support by having the ability to run CP/M application programs.

Like CP/M, Version 1.0 only supported a single directory so all files were seen when the "*dir*" command was used, but several improvements over CP/M were in place. [Ref. 7:p. 634] [Ref. 11:p. 317] The system supported the first five and one quarter inch, single sided disk with a 160 kilobyte capacity. (Version 1.1 later supported double sided 320 kilobyte disks). MS-DOS was the first system to keep the File Allocation Table (FAT) memory resident to decrease disk accesses and increase system speed.

Version 1.0 supported device independent I/O by treating device drivers as files, and used part of the ROM called the Basic Input Output System (BIOS) to house the device drivers. [Ref. 7:p. 634] [Ref. 11:p. 318] Reserved file names are used to call devices, CON for the console, PRN for the printer, and AUX for serial ports. This version also supported shell scripts (a sequence of commands stored in a file) called *batch files*.

MS-DOS Version 2.0 was introduced in March 1983 with IBM's introduction of the PC/XT. [Ref. 7:p. 635] [Ref. 10:p. 129] [Ref. 11:p. 318] Version 2.0 consisted of 20,000 lines of assembly language and several enhancements. The system now supported a ten megabyte fixed disk and 360 kilobyte floppy disks. A hierarchical file system that supported directories, subdirectories and files adopted from Microsoft XENIX made Version 2.0 incompatible with CP/M and established DOS as the dominate operating system for PC's. Another feature like that of XENIX was the ability to redirect inputs and outputs using a pipe (vertical bar), but the function was not as sophisticated as the UNIX based systems.

The popularity of UNIX at the time and the demand for multitasking forced Microsoft to look at the customer base and determine that the majority of the customers were single user on a single system. [Ref. 7:p. 636] Microsoft avoided the cost and overhead of multitasking, but did incorporate *background print spooling* (simultaneous peripheral operations on line). This feature was placed in a command file called PRINT.COM and was designed to run when there was little other DOS activity.

To support new peripherals and keeping with IBM's open architecture, Version 2.0 supported *installable device drivers* (removing then from ROM). [Ref. 7:p. 636] [Ref. 11:p. 319] This was accomplished with the CONFIG.SYS file by using linked lists to join the device drivers. Version 2.05 was designed for international support such as Japanese Kanji characters. Version 2.1 supported IBM PCjr, which was not well accepted by users. Microsoft combined 2.05 and 2.1 into Version 2.11 which was translated into 60 languages and sold world wide. The final versions in this series 2.2 and 2.25 supported Korean and Japanese characters respectively.

Version 3.0 released in August 1984 with 40,000 line of assembly language code was twice the size of Version 2.0. [Ref. 7:p. 637] [Ref. 10:p. 130] [Ref. 11:p. 319] The goals of DOS Version 3.0 in support of the 80286 microprocessor was multitasking, increasing available memory, and networking. Version 3.0 arrived on the market with IBM's new PC/AT, the first PC with the 80286 chip. With 24 address lines the 80286 could access 16 megabytes of memory. [Ref. 10:p. 44] [Ref. 11:p. 336] The use of memory above one megabyte is called *extended memory*. DOS supports utility programs to access this memory.

In support of 8086 and 8088 programs the 80286 has two modes of operation: real and protected. [Ref. 7:p. 637] The *real mode* allows compatibility using a one megabyte address space with 16 bit internal and external data exchange rates (like a fast 8086). The *protected mode* allows for full 16 megabyte addressability with a separate address space for each application program running to support multitasking. Using DOS, only one program is active at any given time.

Version 3.1 released in November 1984 was unique in two ways. [Ref. 7:p. 638] [Ref. 10:p. 130] [Ref. 11:p. 319] First, it was the first time MS DOS and PC DOS were exactly the same and second, this version supported networking. Version 3.1 came with Microsoft Networks to support IBM's PC Network adaptor, a hardware card with a communications controller. The networking capability was limited to "well behaved" application programs. These are programs that are written to not hoard CPU time or use too much memory.

Before the release of Version 3.2 Lotus, Intel, and Microsoft introduced the *expanded memory specification* (LIM EMS) a hardware solution to relax the 640 kilobyte limit on RAM imposed by DOS. [Ref. 7:p. 596] [Ref. 10:p. 44] The specification defined a means for accessing large amounts of RAM by swapping it in 16 kilobyte increments into and out of a 64 kilobyte window in the microprocessors address space. This is usually an unused region of memory between 640 kilobytes and one megabyte. (Popular third party memory managers are Quarterdeck's QEMM or Qualitas's 386Max.)

DOS Version 3.2 supported IBM's PC Network and Token Ring local area networks. [Ref. 7:p. 638] [Ref. 10:p. 131] [Ref. 11:p. 319] This was the first DOS to support three and one half inch sturdy disks with a capacity of 720 kilobytes. The sturdy disk capacity was increased to 1.44 megabytes with DOS 3.3.

Version 3.3 was released in April 1987 with IBM's Personal System/2 (PS/2). [Ref. 7:p. 639] [Ref. 10:p. 130] [Ref. 11:p. 319] The PS/2 was available in various models that gave the consumer a choice of microprocessors (with and without math coprocessors), memory size, and internal bus configuration. The top of the line (at the time) PS/2 Model 80 far exceeded the capabilities of DOS 3.3, which was not a multitasking but a single tasking operating system. Version 3.3 supported real mode for 8086, 80286 and 80386. (The 80386 has 32 address lines capable of four gigabytes of addressable memory). Large memory support was achieved by using 32 megabyte fixed disk partitions. This version also had extensive foreign language support.

Along with the release of the PS/2 came the option for a different operating system all together, namely Operating System /2 (OS/2). [Ref. 7:p. 640] [Ref. 10:p. 170] [Ref. 11:p. 319] OS/2 was supposed to replace MS DOS but late delivery and no graphical user interface gave customers a good reason to stay with DOS.

Version 4.0, released in July 1988, was the first DOS operating system to implement IBM's Common User Access specification. [Ref. 7:p. 660] [Ref. 10:p. 130] [Ref. 11:p. 320] It extended the fixed disk size limit to two gigabytes with LIM EMS and had a vastly improved shell user interface. The shell gave a windows like environment; the File System screen shows a directory tree window and a second window for a list of files in the current directory. The SELECT program allowed the novice to install Version 4.0. The START PROGRAMS screen appears first when the shell is executed. The shell allowed associating data with an application program so that when the data file is selected the application program associated with the data executes.

Microsoft introduced Windows 3.0 in 1990 and DOS 5.0 in April 1991. [Ref. 10:p. 154] [Ref. 11:p. 320] DOS 5.0 with or without Windows 3.0 support was the another attempt to address the 640K limit imposed by earlier releases of DOS. DOS 5.0 was shipped with a two part memory manager called EMM386.EXE and HIMEM.SYS [Ref. 16:p. 317] [Ref. 17:p. 66]. The EMM386.EXE is an extended memory manager that takes advantage of the 80386 architecture to simulate expanded memory using extended memory. It's installed as a device driver in CONFIG.SYS and creates a paging area between 640K and one megabyte to perform the expanded memory function. The paging is in 64K chunks of memory. The HIMEM.SYS is another device driver that allows use of high memory above 640 K. DOS needs both of these drivers to use the high memory area.

DOS 5.0 had two other improvements as well. Removable sturdy disk memory was again improved to support 2.88 megabyte disks. [Ref. 10:p. 130] [Ref. 11:p. 320] The shell was improved to hold several programs in memory at once and support for an extensive help facility. The line editor (EDLIN) was replaced by a screen editor called EDIT. More consideration for testing prior to release was given because this was the first version sold directly to customers.

In the Fall of 1992 Microsoft released Windows for Workgroups (WFW) to support peer to peer networking and a dedicated LAN server [Ref. 10:p. 45,749]. MS DOS Version 6.0 released in April 1993 has supporting software called Workgroup Connection [Ref. 18:p. 64]. MS DOS 6.0 has grown to a whopping eight megabytes if fully installed [Ref. 18:p. 62]. The goal of the Version 6 series is to add utilities to the operating system and deliver a complete package thus avoiding the need for customers to buy additional software products such as data compressors, memory managers and virus scanners. The specifics of these features are discussed in a separate DOS 6 and networking sections.

By November of 1993 Microsoft had an unscheduled release of Version 6.2 to fix some problems with DOS 6.0. [Ref. 18:p. 63] The reason Microsoft named this version 6.2 was to avoid confusion with IBM DOS that was already in release 6.1 [Ref. 19:p. 37]. The major differences between MS DOS and IBM PC DOS is the utilities that are offered [Ref. 20:p. 39]. The basic commands are the same.

Microsoft and IBM had the final code exchange for MS DOS on September 17,1993 [Ref. 21:p. 1C]. Previously, IBM PC DOS would build on code of MS DOS and generally improve it by optimizing. [Ref. 20:p. 39] The standard DOS commands and programming functions are identical in both DOS's but IBM's optimized version has slightly faster character input/output and batch file processing. The user with DOS Version 5.0 can upgrade to either the IBM or Microsoft product.

The third type of DOS is Novell DOS 7.0. [Ref. 20:p. 39] Novell DOS is built upon Digital Research DOS (called DR DOS) that they purchased in 1991. A brief comparison of the three DOS products is addressed in the DOS 6 section. MS DOS is emphasized here because it is what is currently being used at the Naval Postgraduate School.

DOS continues to be the most widely used operating system in the world and continues to grow to meet user demand. Microsoft plans the next release of DOS Version 7.0 to be a 32 bit operating system. The proposed name is DOS-NT (New Technology). [Ref. 22:p. 60] [Ref. 23:p. 1E] The release of MS DOS 7.0 is expected in 1995. Microsoft plans to release Version 4.0 of Windows, code named "Chicago", in 1994.

The chronology of MS DOS major releases are:

* August	1981	Version 1.0
* March	1983	Version 2.0
* August	1984	Version 3.0
* November	1984	Version 3.1
* January	1986	Version 3.2
* April	1987	Version 3.3
* July	1988	Version 4.0
* April	1991	Version 5.0
* April	1993	Version 6.0
* November	1993	Version 6.2

B. DOS PROCESS MANAGEMENT

Although DOS may appear to be multitasking when used with certain application software, it is not a multitasking operating system. The code used to implement DOS is non-reentrant [Ref. 10:p. 134] [Ref. 11:p. 331]. Non-reentrant code must be executed start to finish before a particular module of code may be called and executed again. Unlike UNIX, which can store parameters and keep several processes active, DOS has one active process at a time. DOS does not have the type of *fork* call found in UNIX, and does not have preemptive interrupts based on priorities found in UNIX.

DOS was designed for a *single user on one* computer, executing one task at a time. [Ref. 10:p. 133] [Ref. 11:p. 329] In the DOS shell environment, when the command line is executed, the shell file descriptor is saved and a child process is created. The child process is run to completion. When the child process exits, the shell file descriptors are returned to the parent process. When a process forks off a child, the parent process is suspended. Several processes may be in memory at one time but only one is active. Unlike UNIX, which can return a command prompt before a process is complete (by using "&" after the UNIX command), DOS requires completion of a process before the command prompt is returned.

Processes are spawned by one of three types of executable files. These files are: command, execute, and batch. [Ref.

7:p. 650] [Ref. 10:p. 139] [Ref. 11:p. 328] The file extension ".COM" has no header information and one segment of executable code. The file extension ".EXE" has a header and several segments that are relocatable in memory. The ".BAT" files are a list of DOS commands.

Commands in DOS are either internal or external. [Ref. 7:p. 647] [Ref. 11:p. 323] [Ref. 24:p. 50] The internal commands are in "COMMAND.COM". Internal commands are always resident in RAM. "DIR" and "DEL" for directory and delete are internal commands. External commands are programs stored as files on disk. These commands are transient and are brought in as needed.

All DOS processes begin with a Program Segment Prefix (PSP) which occupies the first 256 bytes of memory for a process. [Ref. 11:p. 329] The PSP contains information about the program such as the file descriptor, program size and address in memory.

Options other than execution for DOS processes are *overlay* and *TSR*. [Ref. 7:p. 647] [Ref. 11:p. 331,337] A running process can load an executable file (.COM or .EXE) called an overlay into RAM and then execute the file. Control is returned to the original running process upon completion. In terminate and stay resident (TSR), after a program is executed it stays in memory without being overwritten. The memory image in RAM is not destroyed. The TSR program is activated by a "hot key".

DOS supports none of the three types of multitasking: cooperative, preemptive and time slice. Multitasking with one processor is just an illusion. [Ref. 10:p. 135] The trick is moving between processes quickly to give the appearance of multitasking. Additional software such as Microsoft Windows and Quarterdeck DESQview may be added to DOS to give a good appearance of multitasking [Ref. 10:p. 137]. The trade off is additional cost and memory overhead. Novell DOS 7.0 uses a "task manager" to accomplish multitasking [Ref. 22:p. 61].

C. DOS FILE SYSTEM and FILE SECURITY

The basis for many of the DOS file system features have their origin with the UNIX file system. [Ref. 10:p. 137] [Ref. 11:p. 148] Both support a hierarchical file system with a root directory, subdirectories, and files. Both recognize current or working directories and the use of relative or absolute path names to access files. One difference is that UNIX uses "/" as path name separators while DOS uses "\" [Ref. 10:p. 216] [Ref. 11:p. 341]. Both support character special files for keyboards, printers and other serial devices and block special files for disks. This is no surprise since Microsoft developed XENIX (a 16 bit UNIX system) prior to DOS.

Differences between the two file systems include : character distinction between upper and lower case, file ownership and security, file mounting and file naming conventions. [Ref. 7:p. 578,601] [Ref. 11:p. 146,15] UNIX distinguishes between upper and lower case letters, DOS does not. UNIX recognizes ownership rights and permissions of a file; DOS does not recognize ownership rights. UNIX uses a mounting system call to attach directories of different file systems. DOS does not supp this type of system call. With mounting available in UNI ie user does not have to worry where in the system the file is located. DOS requires the device name where the file is located like "C:" for a hard drive and "A:" for a floppy disk drive, but this is for files under one file system.

The DOS philosophy holds to the file system, one system for one user. [Ref. 10:p. 142] [Ref. 11:p. 342] DOS supports file attributes which protect the users from themselves but not from other users. The attribute byte will have the following six bits:

* Archive bit - indicates a file was changed,
* Read-Only bit - the file cannot be modified,
* System File bit - reserved so they cannot be deleted,
* Hidden bit - the file is not shown in a directory,
* Volume bit - the entry is a label (name of a disk),
* Directory bit - a directory entry, not a file.

File attributes can be changed or examined with the ATTRIB command. UNIX does not have file attributes like DOS.

Volume labels may be up to 11 characters [Ref. 16:p. 134]. DOS 4.0 and above may also have volume serial numbers. This aids in keeping volumes uniquely labeled.

All DOS directory entries are 32 bytes long. [Ref. 10:p. 142] [Ref. 11:p. 166] It will contain the file attributes, name, and time with date of last modification. The size of the file and the location of the first memory block of the file is also in the directory. From the first block onward the blocks are chained together by pointers.

File naming conventions and file extensions differ considerably between DOS and UNIX. [Ref. 10:p. 138] [Ref.

11:p. 146] UNIX uses either 14 or 255 characters (Berkeley) for file names and does not recognize file extensions. DOS uses the "8.3" convention which means the file name is a maximum of eight characters long (bytes) and the extension is a maximum of three characters long (bytes). UNIX does not recognize or reserve file extensions. Operating systems that support DOS usually shorten a long file name to "8.3".

Extensions which are reserved by DOS are: ".BAT" for batch files, ".COM" for command files , ".EXE" for executable files , and ".SYS" for system files. [Ref. 7:p. 652] [Ref. 11:p. 341] Other common extensions that are not reserved are: ".DAT" for data files, ".DOC" for document files, ".OBJ" for an object file produced by a compiler, and ".TXT" for a text file.

Command files are single segment executable binary files without file header information. [Ref. 7:p. 652] [Ref. 11:p. 341] This means they are absolute machine code that reside in a particular section in memory and cannot be relocated. Command files were designed to load faster than executable files. Executable files are multisegment executable binary file with a header. They are relocatable because the header contains relocation information and the load module. Executable files were designed to facilitate sharing of tasks among procedures so the operating system can support concurrent tasking. To accomplish this the executable code and the data are separated. The executable code has the extension .EXE and by convention the data extension is usually .DAT.

The operating system will normally use either fixed or removable disks to store files. Since most fixed disks are very large they may be divided into one or more partitions. Each partition has a boot section with information about start up and the file allocation table (FAT). [Ref. 7:p. 646] [Ref. 10:p. 140] [Ref. 11:p. 352] The FAT contains information about every block of storage on the disk. Each FAT entry is a 16 bit data word that indicates one of four conditions about the block. The block may be bad (unusable), it may be free (usable), it may be an end of a file (EOF) or the entry has the address of the next chain of blocks that make up a file (a pointer). The basis for the DOS file system is the file allocation table (FAT).

To create, open, or close a file or device a special 16 bit integer called a *file handle* is used (like the file descriptor in UNIX) [Ref. 11:p. 15]. [Ref. 7:p. 650,651] Earlier versions of DOS used a file control block (FCB) which was 37 bytes of data that was stored with an application program and contained file information. Although both FCB's and file handles are supported, file handles are used almost exclusively because they support file locking necessary for file sharing. File handles are also necessary for a hierarchical file system that DOS supports.

When a file is opened the path name and attributes are passed to DOS which in turn creates the file handle and passes the handle to the requesting program. [Ref. 7:p. 635] [Ref. 11:p. 159] The program saves the file handle to specify operations on the file. DOS creates a table to relate file handles to files or devices. Normally eight file handle entries are assigned by default, but this can be increased to 20 in CONFIG.SYS.

The file system is always tied closely to input and output. When a process starts in UNIX three files are opened: standard input, standard output and standard error. When a process starts in DOS five files are opened. [Ref. 7:p. 652] The first three are the same as UNIX, the other two are for a serial communication line and a printer. These five file handles are assigned entries zero through four.

Device drivers are treated like files in DOS. [Ref. 11:p. 343] This is very convenient because users can add a single line to CONFIG.SYS to install a new device. The statement added to CONFIG.SYS is the path name of the file containing the device driver. A common device driver for graphical user interface is MOUSE.SYS for the mouse.

Some of the first drivers developed for DOS were RAMDRV.SYS and SMARTDRV.SYS. [Ref. 11:p. 343] RAMDRV.SYS is a RAM disk driver for extended memory. SMARTDRV.SYS is the disk cache driver used in extended memory. Extended memory is the memory above one megabyte that DOS normally will not access [Ref. 11:p. 336]. Expanded memory is a mapping mechanism used for making more memory available to a program than would normally be the case [Ref. 11:p. 338].

The DOS 640K limit has received so much attention that memory managers load device drivers into upper memory. This allows more space in conventional memory below 640K. DOS 5.0 also introduced two new device drivers for memory management that work together. [Ref. 16:p. 287,288] EMM386.EXE and HIMEM.SYS are for use on 80386 CPU's and higher. They can take advantage of extended and expanded memory.

File security in DOS is weaker than in UNIX, where it is built-in. DOS is a single user system that does not require log on and provides little protection for files from unauthorized use. [Ref. 10:p. 143] Hardware protection or an additional layer of software is required for security. A file may be hidden but several types of utility programs can find and retrieve the hidden file.

D. DOS INPUT/OUTPUT SYSTEM

Device drivers provide the basic input and output (I/O) for DOS. [Ref. 7:p. 652] [Ref. 10:p. 67] [Ref. 11:p. 342] DOS supports device drivers which are hardware specific programs that interface with the DOS operating system. The manner that device drivers communicate to the device is transparent to DOS because all devices communicate with DOS in a standard way. The two types of device drivers are *character device drivers*, that handle I/O serially, one character at a time, and *block device drivers*, that handle randomly accessible blocks from fixed and removable disks. These device drivers are stored as character and block *special files*.

These device drivers are called special files because they are part of the file system. [Ref. 11:p. 343] DOS keeps all device drivers in CONFIG.SYS. To add a new device the user needs only to add a line to this system file. The command is *DEVICE = name of device driver* [Ref. 16:p. 251]. In contrast, the UNIX system administrator normally keeps device drivers in a special directory called "/dev" (short for device). The system administrator is normally the only user to add or delete devices, which have to be linked with the UNIX operating system.

It is important to note that the device driver names are reserved words in DOS and may not be used as a file name regardless of extension. DOS maintains a linked list of devices and unlike UNIX (that sometimes uses embedded drivers) requires no special linking to the operating system [Ref. 11:p. 343].

Device drivers have three parts: a device header, a strategy routine, and an interrupt routine. [Ref. 7:p. 653] The device header points to the next device in the linked list chain of devices. It also has pointers to the strategy routine and the interrupt routine. When a device is called the pointers linking the device to the call are stored by the strategy routine and the actual input or output is handled by the interrupt routine that returns the status and completion information to the strategy routine. The interface between these routines and the calling program is a data structure called a request header.

The five files automatically opened for a process I/O in DOS are actually device drivers with file handles zero through four. [Ref. 7:p. 652] They are :

* 0 standard input device (keyboard),
* 1 standard output device (display),
* 2 standard error device (display),
* 3 standard auxiliary device (serial port),
* 4 standard printer device (parallel port).

Names given to these device drivers are console (CON), auxiliary (AUX) and printer (PRN).

DOS I/O can be redirected as in UNIX using "<" for input and ">" for output to a file. [Ref. 10:p. 146] [Ref. 11:p. 275] [Ref. 16:p. 161-163] The double greater ">>" will append output to a file without destroying the previous contents. DOS also supports filters and the pipe symbol "|" similar to UNIX. Filter functions are: *find* to search for a character string, *sort* to sort text data, and *more* to display one screen of data at a time. The pipe allows output from one program to be input to the next, handled in left to right order.

E. DOS SHELL USER INTERFACE

The MS DOS shell has been available since Version 4.0. [Ref. 10:p. 150] [Ref. 11:p. 324] The shell is an alternative to the command line interface that typically prompts the user with the active disk drive such as "C:\". The DOS shell is a screen oriented interface that allows users to point and click with the mouse to carry out commands. The shell is by no means an elaborate graphical interface like Microsoft Windows.

The shell allows the user to switch between tasks without stopping and restarting but the tasks do not run in the background like UNIX daemons. [Ref. 10:p. 153] Programs are suspended when more than one program is active. Certain tasks, such as setting time and date, still must be accomplished using the command line. There is no command prompt for DOS shell, but command prompt can be selected at the bottom of the screen by selecting "Shift and F9".

The DOS shell is divided into five areas. [Ref. 10:p. 150] [Ref. 11:p. 325] [Ref. 16:p. 31] The major areas of the shell are:

* Title Bar,
* Menu Bar,
* Directory Tree area,
* File List area,
* Program List area.

The top of the graphical shell layout is a title bar that identifies the screen as the "DOS Shell". Shells from third party vendors are also available [Ref. 11:p. 325].

Directly below the title bar is the menu bar that lists pull-down menus that access certain tasks. [Ref. 10:p. 152] [Ref. 16:p. 30] The menu bar offers choices of "File", "Options", "View", and "Help" (with Versions 5.0 and above). Directly below the menu options the disk drive icons (letters) are displayed. The active drive is highlighted. The "File" option under the menu bar controls File Copy, File Rename, File Delete, and File-Create Directory. [Ref. 10:p. 152] [Ref. 16:p. 30] Other file functions allow for changing file attributes and viewing file contents. (File comparison is not directly supported by DOS Shell, but is available through the command prompt.)

The "Options" selection changes screen colors and allows certain options to be disabled, such as asking for confirmation before completing a task.

"View" controls size and display of the four available windowing sections: Directory Tree, File List, Main, and Active Task List.

The Directory Tree, below the menu bar, is the graphical representation of the current (or working) directory structure. [Ref. 10:p. 150] [Ref. 16:p. 29] The File List, to the right of the Directory Tree, shows the files in the current directory.

The Program List area which is below the Directory Tree and File List can be divided to display a *Main* window area and an *Active Task list* window area.

Main lists commonly used programs and typically has four program entries. The first program entry is "Command Prompt" that accesses the command line processor. [Ref. 10:p. 150] [Ref. 16:p. 29] The second entry is "EDITOR" which is a screen editor unlike the old "EDLIN" used in early DOS Versions. The third entry is "BASIC" that activates the BASIC programming language, and the last is "Disk Utilities" that controls disk functions for formatting, dumping and restoring disks.

Active Task list is available on DOS Version 5.0 and higher [Ref. 16:p. 31]. DOS 6.0 has five key combinations to assist the user in managing the Active Task list. [Ref. 24:p. 14] The key combinations used with multiple tasks are:

* "Shift and Enter" adds a program to the list.
* "Alt and Esc" switches from one active task to the next.
* "Shift and Alt and Esc" switches to the previous active task.
* "Alt and Tab" switches between two active tasks.
* "Ctrl and Esc" switches between the active program back to the DOS shell.

The DOS shell is a very limited single tasking graphical interface. Microsoft currently offers Windows 3.1 to provide a true graphical interface to DOS. Windows is an operating system environment program for DOS. Windows requires DOS to be installed before it can operate. Windows is the most common enhancement to the DOS interface, but third party graphical interfaces for DOS are also available.

F. WINDOWS 3.1

Microsoft Windows 3.1 (introduced in 1992) is a popular alternative user interface for DOS-based PC's. Windows 3.1 is not an operating system. It may be best described as an operating system environment that allows multitasking for DOS application programs [Ref. 25:p. 64].

Windows 3.1 requires DOS in order to operate. [Ref. 10:p. 154] [Ref. 26:p. 64] Windows 3.1 is a large program consisting of over two million lines of code [Ref. 27:p. 3D]. Windows 3.1 is a true graphical interface and not just a DOS shell. It is important is not to confuse Windows 3.1 with WindowsNT which is a 32 bit, self-contained, DOS independent, operating system [Ref. 28:p. 28]. The next upgrade to Windows 3.1 will be Windows 4.0 (also called Windows Chicago). Windows 4.0 is expected to be released in 1994.

Windows (henceforth meaning Windows 3.1) supports two types of application programs, DOS and Windows. For example, the word processing program "WordPerfect" is available for DOS and Windows as separate purchases. The user can switch between the Windows environment to the DOS environment to run programs directly in a "DOS mode". This is convenient if the user has the DOS copy of an application program and not the Windows version.

Windows can run in a 386 enhanced mode that supports multitasking for DOS-based application programs. [Ref. 10:p.

160] When Windows is running in the 386 enhanced mode it provides "user defined" preemptive multitasking for DOS application programs. (This is not considered true multitasking like UNIX.) The user can assign available RAM and a percentage of CPU time to a DOS session. This type of multitasking is called "time slice" since the user can assign processing time.

The Windows "386" enhanced mode creates a virtual 8086 environment for each DOS session. [Ref. 10:p. 160] When an I/O is requested by a DOS session the Windows version of MS DOS intercepts the I/O and simulates an 8086 DOS environment.

The user running DOS without Windows can switch between programs using key combinations. When switching between application programs under DOS, three events occur: the switched program is stopped, the program parameters are stored, and a new program begins. Only one active program is running at a time. In the "386" enhanced mode the user can run several DOS sessions with each receiving some processing time.

Third party vendors provide additional features for the DOS and Windows combination. For example, Firefly Software produces software called Enhanced DOS for Windows or EDOS. [Ref. 26:p. 64] This product allows multiple DOS sessions to control up to 736K of RAM each. EDOS also provides an interface between the Windows clipboard and the DOS prompt for data transfers.

Windows supports "cooperative" multitasking for Windows application programs. Cooperative multitasking is non-preemptive. [Ref. 10:p. 159] These programs are "well behaved" programs that share CPU time cooperatively without user intervention. Application programs written especially for Windows yield control of the CPU at regular intervals. Each program running must keep track of CPU time used and yield control if another program needs time. Cooperative multitasking is also found with OS/2 and System 7 applications.

The Windows graphical interface can perform system commands in two areas: the File Manager and the Program Manager. [Ref. 10:p. 154] The File Manager is very similar to DOS Shell 5.0 in functionality but with more overhead for the graphical interface used with Windows. The File Manager is actually started from the Program Manager control window. The File Manager can associate data with an executable file (program). This way, input data is readily available. The Program Manager gives the user five ways to start a program [Ref. 10:p. 159]. This allows for mouse and keyboard commands.

Windows can take advantage of the version of DOS installed with it. In 1993 DOS 6.0 was released bundled with several utilities. The version of DOS used is very important for Windows because some programs incorporated into DOS 6.0 are specifically for Windows. DOS 6.0 Antivirus, Backup and Undelete are Windows oriented programs along with a Windows hosted compression program that works with DOS 6.0 DoubleSpace. (Details of these utilities are in the DOS 6.0 and DOS 6.2 section.)

Unlike running DOS alone, the addition of Windows to DOS requires an 80386SX at a minimum and a recommended four megabytes of RAM [Ref. 10:p. 124]. Windows actually needs eight to 12 megabytes of RAM to run multiple sessions effectively. DOS 6.0 with Windows 3.1 together requires 15.8 to 18.3 megabytes of fixed disk space [Ref. 29:p. 1E]. DOS 6.0 alone requires six to eight megabytes of disk space.

The Windows graphical interface overhead for application programs has a much greater demand for fixed disk space than DOS applications. [Ref. 29:p. 1E] [Ref. 30:p. 1000] WordPerfect 5.2 for Windows requires up to 12 megabytes of fixed disk space while WordPerfect 5.1 for DOS requires between 2.5 and 4.5 megabytes of fixed disk space.

The DOS with Windows combination continues to out sell the primary competitor for the same market, IBM's OS/2. Windows has the same processor requirements and about the same memory requirements as OS/2.

G. DOS 6.0 and DOS 6.2

Microsoft released DOS 6.0 in April 1993 with a host of new features. [Ref. 18:p. 62] [Ref. 31:p. 347] These features fall into the general categories of: communication, protection, configuration and memory. For the first time DOS comes as an integrated package, alleviating the need for several add-on programs. The major competitors in the DOS market, IBM and Novell, also have new features worth mentioning.

DOS 6.0 comes equipped with a zero slot communication program called INTERLNK. [Ref. 17:p. 66] [Ref. 24:p. 788] The zero slot concept means that a network interface card (NIC) is not required for communications. One PC must have DOS 6.0 or higher installed and another PC must have DOS 3.0 or higher installed for the two to communicate. The PC's can communicate over parallel or serial ports. (This has long been a feature of Macintosh LocalTalk cabling.) The primary use for INTERLNK is to down load a laptop client to a home server PC. The laptop has the INTERLNK.EXE program installed, while the server runs a program called INTERSRV. The RAM requirements are minimal. At least 130K of RAM is required for the server and 16K of RAM for the client.

DOS 6.0 has three areas of protection: user, backup, and virus. To protect users from themselves, the UNDELETE command attempts to retrieve a file that has been deleted. UNDELETE actually hides the deleted file until the memory is needed. If the file has not been overwritten it may be retrieved. [Ref. 24:p. 758] [Ref. 32:p. 52] This is not fool proof, but it does allow for retrieval if the mistake is caught early. The best way to protect files is backup. The MSBACKUP command provides a means of backing up a fixed disk on a series of removable disks [Ref. 18:p. 66] [Ref. 24:p. 385]. Using the DOS shell, these programs are located under the DOS Utilities program group of the Main area.

Since computer viruses continue to plague users, DOS 6.0 has incorporated the Anti-Virus facility. [Ref. 18:p. 63] [Ref. 24:p. 292] Two types of virus protection are offered with a scanner and a shield. MSAV is the DOS scanner that reports and flags bad checksums then attempts to clean them up. Vsafe is the virus shield. Vsafe currently has two problems. One problem is the Vsafe alerts on TSR programs (programs that terminate and stay resident in RAM). These false alerts may be annoying to users. The second problem relates to the first. Users may remove Vsafe with a keystroke to avoid false alerts. Removing the virus shield negates the purpose of having it installed.

MWAV is the Windows equivalent program to MSAV. [Ref. 18:p. 63] [Ref. 24:p. 798] The Anti-Virus facility is found under the Microsoft Tools in the Windows Program Manager. The additional memory and RAM required for Windows programs makes

RAM management and fixed disk space important issues.

The 640K barrier continues to be a major drawback to DOS. DOS 5.0 addressed the issue with EMM386.EXE and HIMEM.SYS [Ref. 16:p. 317]. These memory managers fell short in actual program placement and squeezing. [Ref. 33:p. 393] [Ref. 34:p. 6F] DOS 6.0 addresses one of these functions with MemMaker. MemMaker can analyze a program and place it in upper memory blocks much better than DOS 5.0. However, DOS 6.0 cannot squeeze a large program into upper memory like Qualitas 386MAX 7.0 and Quarterdecks QEMM386 7.0. These third party products are considered complete memory managers. Both third party memory managers cost about $100.

Relating closely to RAM management is the issue of system configuration, (installing device drivers and assigning the number of files and buffers). DOS 6.0 provides the MULTICONFIG which permits several system configurations. [Ref. 18:p. 64] [Ref. 24:p. 307] It will pause at CONFIG.SYS command and ask for user conformation to execute the comman This permits a selection of device drivers to be installed and the number of files and buffers assigned which saves on RAM usage.

MULTICONFIG will allow the user to choose portions of CONFIG.SYS and AUTOEXEC.BAT files for start up. [Ref. 24:p. 310] Depending on the start up selection different sections of CONFIG.SYS and AUTCEXEC.BAT will be executed. For example, the user may select: a "Minimum DOS Installation", a "Normal DOS/ Windows Installation", or an "Optimized for Windows Installation". This selection will determine commands executed in the files. Default settings can also be used with MULTICONFIG so user intervention is not needed. MULTICONFIG alleviates the need for a clean boot disk when start up files are corrupted because several configurations are available.

To fix a fragmented fixed disk (which degrades performance) and save fixed disk space DOS 6.0 has the DEFRAG command and the DoubleSpace compression program. [Ref. 24:p. 717] [Ref. 35:p. 54] The structure of the File Allocation Table (FAT) causes fragmentation when files are removed or changed. DEFRAG will put a fragmented file back together. The added benefit is that the UNDELETE command will probably work better for recovery because the file is not fragmented and is easier to retrieve. DEFRAG requires a large area of RAM to accomplish its work, so TSR programs should not be present. Before compressing the files the DEFRAG command should be used to combine the fragmented files.

Compression algorithms were previously only available from third party sources. Now compression is a standard part of DOS 6.0, via DoubleSpace. [Ref. 24:p. 320] This gives the user more room on the fixed disk, but usually a slower response time since compression and decompression is accomplished on-the-fly.

The initial release of DoubleSpace had a problem. DoubleSpace was writing data to fixed disk areas prior to checking if the area was good or bad. To correct the problem Microsoft replaced the ChkDisk program with ScanDisk. This new program ensured the disk was checked prior to a write. Consequently, Microsoft had an unscheduled release of DOS 6.2 in November 1993. [Ref. 31:p. 347,349] MS DOS 6.1 was not released because IBM PC DOS was already in release version 6.1. MS DOS 6.2 was released to avoid confusion with IBM PC DOS [Ref. 19:p. 37]. DOS 6.2 has two additional features as well, the ScanFix program and the UNCOMPRESS command. [Ref. 19:p. 37] ScanFix is a utility for general disk repair to decrease the number of bad disk blocks. UNCOMPRESS is a command that disables DoubleSpace. Another simple solution is not to install DoubleSpace.

IBM and Microsoft had the final code exchange for MS DOS on September 17, 1993 [Ref. 21:p. 1C]. Prior to this IBM took the Microsoft code and optimized it. [Ref. 20:p. 39] Since IBM sells hardware some of the components are also optimized for PC DOS. The major difference between MS DOS 6.2 and IBM PC DOS 6.1 are the utilities packaged with the operating system. The basic DOS commands are the same for both operating systems. IBM's compression program is SuperStore. IBM uses Central Point Backup for DOS and Windows. IBM uses an in-house virus protection program originally written for large corporate accounts.

The third competitor in the DOS market is Novell DOS 7.0. [Ref. 20:p. 40] [Ref. 36:p. 78] Novell, best known in the networking arena, purchased DR DOS from Digital Research in 1991. The latest release is an upgrade from DR DOS 6.0 called Novell DOS 7.0. This new version incorporated Stacker disk compression as part of the operating system. A "Task Manager" is used to accomplish what Novell calls preemptive multitasking (not just task switching). Novell DOS 7.0 supports the DOS Protected Mode Interface (DPMI) that allows built-in utilities to run in extended memory instead of upper memory blocks.

Regardless of which DOS is used, they are all limited by the fact that they are 16 bit operating systems. All of the DOS products go to extremes to alleviate the 640K barrier.

E. DOS and WINDOWS NETWORKING

Microsoft's first endeavor in DOS based network support was with DOS 3.1. It ran with Microsoft Networks Operating System (NOS). [Ref. 7:p. 638] [Ref. 10:p. 152] [Ref. 11:p. 319] Next DOS 3.2 was released to support IBM Token Ring local area network. DOS has traditionally relied on a server and a Network Operating System such as IBM's DOS based PC LAN program. More recent NOS's for DOS are OS/2 based such as Microsoft's LAN Manager and IBM LAN Server [Ref. 10:p. 52]. LAN Manager and LAN Server are covered separately. Novell's Netware continues to be a strong competitor in the LAN market with DOS support, but is beyond the scope of this thesis. [Ref. 10:p. 62].

With the release of DOS 6.0 the system now supports peer to peer (one to one) file transfer with INTERLNK and client server (one to many) networking with Workgroup Connection [Ref. 18:p. 63]. This is in addition to Microsoft Add-on for MS DOS that allows non Windows PC's to network [Ref. 37:p. 38].

INTERLNK was discussed with DOS 6.0 so only a brief review is given here. The INTERLNK program allows selection of client or server between two DOS systems. [Ref. 24:p. 789] One system must be running DOS 6.0 and the second system must be running DOS 3.0 or higher. The peer to peer connection can be over the parallel or serial ports. The objective of

INTERLNK is the ability to load data from a laptop computer to the home based PC or vice versa [Ref. 18:p. 64]. INTERLNK is installed as a device driver under CONFIG.SYS [Ref. 18:p. 64] [Ref. 24:p. 790].

DOS support in a client-server environment may be accomplished using Windows for Workgroups (WFW). The latest release is WFW 3.11 (October 1992). [Ref. 18:p. 64] [Ref. 37:p. 38] DOS 6.0 (November 1993) is the first release since the WFW upgrade. DOS 6.0 offers a complementary software package to WFW 3.11 called Workgroup Connection. Workgroup Connection is not a stand alone product; it runs in conjunction with a server that has WFW running.

WFW supports DOS and Windows based PC's. Separate add-on packages for WFW support is available for both. [Ref. 37:p. 38] Workgroup Add-on for Windows 3.1 costs about $70 and an add-on for MS DOS 3.3 or higher costs about $50. The WFW 3.11 complete package costs about $230, exclusive of network card cost. WFW claims to support 107 NIC's [Ref. 10:p. 750].

WFW permits groups to be set up for file and printer sharing. [Ref. 10:p. 751-763] [Ref. 37:p. 39] The base protocols are completely compatible with another Microsoft product, LAN Manager. WFW supports networking with WindowsNT and Novell networks using the International Packet Exchange (IPX) protocol. WFW also supports TCP/IP.

I. DOS ADVANTAGES and DISADVANTAGES

DOS is the most popular operating system worldwide and has, at least, 20,000 application programs [Ref. 7:p. 629]. DOS application programs are almost always less expensive than programs for other operating systems. DOS has wide spread device support. [Ref. 10:p. 162] Almost every add-on device has drivers to support DOS. DOS is the simplest and least expensive of the major PC operating systems and has minimal hardware requirements. DOS can run on the 8088 processor and with less than one megabyte of RAM.

Great strides have been made to improve DOS since its first release in 1981, but old problems still persist. Newer versions of DOS provide an on line HELP facility and a limited, but effective graphical interface. One of the greatest limitations to DOS, the "640K" barrier has been addressed by hardware and software to solve the problem. DOS continues to be backwards compatible with the 8086 original CPU architecture, which has limited memory addressing capability. DOS 5.0 and DOS 6.0 both offer solutions to the 640K barrier.

Another problem with DOS is that it is a single tasking system. DOS needs Windows for multitasking and Windows needs DOS to run. To perform multitasking DOS is cradled in the Windows 3.1 program. Several DOS sessions may be simulated in a virtual 8086 mode called the 386 enhanced mode. This has two major drawbacks. Windows needs at least an 80386SX to run effectively and four megabytes of RAM is recommended. Windows is an additional software cost. The additional hardware and software costs and overhead defeats the advantage of DOS minimal requirements. DOS is a 16 bit operating system that cannot take full advantage of a 32 bit CPU.

DOS is sufficient for one user running one program at a time. Unless the DOS user needs to run more than one application at a time the Windows advantage of multiple sessions is lost. For example, if the user wants to run a word processor and a spread sheet at the same time and switch between the two Windows is a good choice. However, because most programs require frequent user intervention to operate, such as typing in commands and initiation of print jobs, it is difficult to achieve concurrent execution for multitasking. The user may just as well use a fast task switcher like WordPerfect Shell.

If the user is running a single application (such as a mathematics program) the program will normally run faster in the DOS environment by bypassing the extra layer of Windows software. Many Windows users run DOS programs on Windows because the Windows version is not available or they already have the DOS version and do not want to buy the Windows version.

Some programs are written for Windows and are not available for DOS. The TrueType fonts program presently is available for Windows only. TrueType stores the image of characters and not as bit map like the Adobe fonts (used with OS/2). Many other programs written for Windows are made quite efficient by using cooperative multitasking techniques.

Version 5.0 was the last DOS that IBM and Microsoft jointly supported. Traditionally the MS DOS source code was given to IBM under contract. IBM would optimize the code and release its own version, PC DOS. After September 1993 the DOS code exchange stopped and Version 6.0 from both companies supported different utilities. The user can upgrade to either IBM Version 6.1 or Microsoft Version 6.2 from Version 5.0 and gain an advantage from the new utilities and improved user Shell.

DOS has no security provisions unless extra software is added. Security is provided by LAN server software, for network applications. DOS-based network operating systems, like IBM's PC LAN, provide read only and generic password security of server files.

The DOS user may write a simple program (batch file) to blank the screen when not in use and have the screen return with a password. This is a very limited feature that is not too difficult to break. With the government requirement for C2 level security (log on, file permissions and encryption) the additional cost in providing security for a DOS environment should be considered. Peer to peer and limited client server communications supported by DOS 6.0 shows a departure from the DOS single user environment and a greater need for security.

The usage environment should determine the need to use DOS. The single user with a single system performing a single task at a time is the environment DOS was originally created to support. DOS is a good choice for simple operating system and it is attractively priced. Application programs for DOS are less expensive and require less storage space than their GUI counterparts.

The unscheduled release of DOS 6.2 to fix DOS 6.0 is an example of how new releases of software may have bugs. Some users complained of problems with DoubleSpace, which came with DOS 6.0. DOS 6.2 corrected the problem. New software releases (ending in zero) may need to be treated more skeptically than later releases.

IV. OS/2

A. OS/2 BACKGROUND

IBM and Microsoft began the project for a 32 bit personal computer operating system in 1984. [Ref. 7:p.785] [Ref. 10:p. 170] The initial goal was to develop a general purpose multitasking system that could take advantage of the 80286 protected mode. The operating system name was changed from DOS Version 5.0 to CP/DOS and eventually to Operating System 2 or OS/2. OS/2 became the first thirty two bit operating system for PC's.

OS/2 was to replace DOS by breaking the DOS "640K" barrier and by providing true multitasking with a graphical user interface. [Ref. 7:p. 640] [Ref. 10:p. 170] [Ref. 38:p. 93] The first release of OS/2 was well behind in schedule and functionality. Version 1.0 released in 1987 was doomed to failure under the 80286 architecture. The 80286 could not support the multitasking requirements of the operating system because the processor could not switch freely between the real mode for standard DOS programs and protected mode needed for multitasking. IBM soon realized the minimum processor power needed for OS/2 was the 80386.

Other problems with Version 1.0 was that it had no graphical interface and thus no mouse support. It required four megabytes of RAM. [Ref. 10:p. 170] [Ref. 38:p. 93] This memory requirement was very expensive at the time. A memory shortage had driven market price up to $500 per megabyte. Version 1.0 could only address thirty two megabytes on a fixed disk.

IBM previously released two editions of any OS/2 version, a *standard edition* and an *extended edition*. [Ref. 7:p. 641,785] [Ref. 9:p. 338] Standard edition 1.0 was just the basic operating system. It supported multitasking with a command line user interface. Data communications was similar to PC DOS. Extended edition 1.0 is for networking with a mainframe host and included a *communications manager* for the 3270 mainframe terminal emulation. Extended edition 1.0 also supported a *database management system* (DBMS) and included the IBM LAN Server (NOS) with enhanced data communications. Neither the standard nor the extended edition had a graphical user interface (GUI) in Version 1.0.

Over the next three years (1988 to 1991) improvements to both editions resulted in three new releases. [Ref. 10:p. 170] In November 1988, OS/2 Version 1.1 was released with a GUI called the *Presentation Manager* for both the standard and extended editions of the operating system. One year later Version 1.2 was released with the new High Performance File System (HPFS) and a new batch command language called REXX. This version of OS/2 was the first to support multiple DOS sessions. Early in 1991, OS/2 Version 1.3 was released. IBM had optimized the code and added built-in support for the new font technology at the time called the Adobe Type Manager (ATM). ATM stores characters as a bit map image. Despite improvements OS/2 received only a very limited market share.

IBM at one time was trying to optimize the OS/2 code for the PC/2 with the Micro Channel internal bus structure. [Ref. 11:p. 319] [Ref. 38:p. 93] Sluggish releases of OS/2 and a potentially limited market prompted Microsoft to drop the joint OS/2 project in 1991. (This allowed Microsoft to devote more resources to MS DOS, Windows, WindowsNT and other software products.) The final source code exchange between Microsoft and IBM for DOS and Windows was September 1993 [Ref. 21:p. 1C].

"DOS better than DOS" and "Windows better than Windows" was IBM's sales pitch for the release of OS/2 Version 2.0 in March 1992 (originally scheduled for release in April 1991). [Ref. 10:p. 170] [Ref. 39:p. XXIV] The GUI was changed to the Workplace Shell that supports a more object oriented session.

OS/2 Version 2.0 supports DOS and Windows 3.1 application programs. Multiple sessions of both types of applications can be run under a DOS session and a WIN-OS/2 session. [Ref. 10:p. 194,195] [Ref. 39:p. 66,67] OS/2 Version 2.0 includes a modified version of Microsoft Windows embedded in every copy. OS/2 has its own set of 32 bit application programs in addition to the DOS and Windows 16 bit applications.

Whether 16 bit applications run better under the OS/2's 32 bit architecture is controversial. For every article that states OS/2 runs these applications faster, another article will say the DOS or Windows native 16 bit environment is more efficient. An extra layer of software will generally slow down an application. OS/2 provides an emulation environment for DOS and Windows so an extra layer of software is present [Ref. 39:p. 67]. OS/2 does provide the flexibility of 16 or 32 bit applications not offered by DOS or Windows 3.1.

Some DOS applications do not run under OS/2 because they use low-level hardware or non-standard memory access. [Ref. 10:p. 194,195] [Ref. 39:p. 67] The DOS application programs that usually will not run under OS/2 emulation generally fall into five categories: disk diagnostics, communications, games, graphics and programs that support advanced microprocessor features of 80386 or 80486 architecture.

September 1993 marked the release of OS/2 Version 2.1 with full functionality. [Ref. 40:p. 3] The system offers a boot manager that gives a choice of operating systems to start. OS/2 also offers crash protection so a single session will not stop the entire system. OS/2 comes bundled with a database manager, a spreadsheet and editors for sound and text. The memory and CPU requirements are no higher than for Version 1.0.

OS/2 support of multiple sessions of application programs in a multitasking environment depends on the amount of available RAM. [Ref. 10:p. 168] A basic rule of thumb for RAM requirements is four megabytes for OS/2 plus one megabyte for each DOS session plus two megabytes for each Windows session running simultaneously.

The fixed disk memory requirements for OS/2 is between 15 to 30 megabytes of fixed disk space depending on what functions are installed [Ref. 10:p. 168]. OS/2 Version 2.1 is installed using over 20 magnetic disks or two CD ROM disks [Ref. 38:p. 93]. OS/2 has become so large that IBM is contemplating a new version called OS/2 Lite.

OS/2 is not the only 32 bit operating system being marketed. In addition to UNIX products already available, Microsoft released a new product ,WindowsNT, in August 1993. The memory requirements for UNIX and WindowsNT are twice that of OS/2. WindowsNT requires an 80486 to operate. OS/2 can run on the 80386 at a minimum of 16 megahertz.

Unlike Microsoft and other software vendors, IBM provides free technical support for OS/2 by telephone [Ref. 40:p. 4]. IBM's CEO, Louis Gerstner, continues to stand behind OS/2 despite lower than expected sales and greater than expected investment. The design of OS/2 is under IBM's System Network Architecture (SNA) which ensures compatibility with some of IBM's other products. (IBM's DOS-based, PC LAN program is not compatible with OS/2.)

Milestones for OS/2 releases are:

* 1984	Project begins
* 1987	Version 1.0
* 1988	Version 1.1
* 1989	Version 1.2
* 1991	Version 1.3
* 1992	Version 2.0
* 1993	Version 2.1

B. OS/2 PROCESS MANAGEMENT

OS/2 has three modes of operation: the OS/2 mode,the DOS mode and the WIN-OS/2 mode (for Windows). [Ref. 39:p. 68] Any mode that is running on the computer is called a session. To support these modes, OS/2 takes a unique approach to process management. The outcome is a process management system that resembles UNIX and Microsoft Windows with unique features of its own.

Similar to UNIX, OS/2 recognizes the concept of an executable path through a program called a thread. [Ref. 7:p. 790] [Ref. 10:p. 172] [Ref. 39:p. 235] OS/2 treats a program as one or more threads. The advantage of thread processing is the decrease in overhead because threads share the resources of a process. OS/2 has the unique ability to allow the user to set a maximum number of active threads in a process.

Like UNIX, OS/2 assigns each process a "Process Identification" number (PID) to keep track of a process and its threads. [Ref. 7:p. 788,790] Each process has its own address space. OS/2 ensures the address spaces do not conflict between processes. OS/2 uses RAM semaphores to test and set shared memory within a process.

Communications between processes are facilitated by using pipes, queues and signals that work the same as their UNIX counterparts. [Ref. 7:p. 799-802] [Ref. 40:p. 16] Pipes that allow interprocess communication are First In First Out (FIFO) queues. Interprocess communication using queues may be randomly accessed because a pointer to memory is passed. Signals are software interrupts that direct the process to interrupt handling software. Processes may or may not act upon a signal. If more than one signal is sent to a process they are handled in FIFO order or ignored. Signals have no priorities, but processes are given three levels of priority in OS/2.

The priority system used in OS/2 processes differs from that used in UNIX. Like other priority-based operating systems, OS/2 will run the highest priority process (or thread) that is in a ready to run state. OS/2 gives a higher priority to response time to the user than to system throughput [Ref. 10:p. 173].

The three classes of priority (highest to lowest) are: *interactive*, *foreground*, and *background*. [Ref. 7:p. 791] [Ref. 10:p. 173] "Interactive" gives the user an immediate response which is the objective of OS/2. Active programs receive both interactive and foreground priorities. "Foreground" priority is always applied to the visible screen (or session) where the user is doing work. "Background" priority is assigned to the non-visible screens and receives CPU time only when the interactive and foreground priority programs are not busy.

An example of the OS/2 priority system in a multitasking environment is to download from a bulletin board system (BBS), format a floppy disk, and work on a word processor. [Ref. 10:p. 173] The BBS download and the disk format are handled in the background while the word processor is in the foreground. A system error needing interactive response will take priority over the foreground word processor. The background processes are given CPU time when the word processor is not busy.

Like UNIX, OS/2 will give I/O bound processes a higher priority then CPU bound processes to ensure a good device utilization, but a background process will never have a higher priority than an interactive or a foreground process [Ref. 7:p. 793].

OS/2 is considered a true preemptive multitasking operating system, like UNIX, because priorities are calculated and acted upon without user intervention. Unlike UNIX, OS/2 will also give the user commands to control multitasking in a time slice manner. To support Windows applications cooperative multitasking is also supported.

The user can set the maximum number of threads a session can create at one time. The command "THREADS = xxx" has a range from 32 to 4095; the default is 64 [Ref. 39:p. 235]. The user can also set the time slice for DOS and OS/2 sessions using "TIMESLICE = minimum time, maximum time". [Ref. 39:p.

236] The times are in milliseconds and the minimum time must be at least 32. The maximum time is optional, but if used it must not exceed 65,536.

The "PRIORITY" command applies to all threads within a process. [Ref. 39:p. 226] The "PRIORITY" setting is either "DYNAMIC" or "ABSOLUTE". "DYNAMIC" is the default setting and allows the priority to change. "ABSOLUTE" ensures a priority stays the same. Dynamic priorities may be further controlled with the "MAXWAIT" command. "MAXWAIT" is measured in seconds and establishes a length of time a process must wait until the priority is changed. [Ref. 39:p. 222]. The range is one to 255 with no default value.

The DOS mode and WIN-OS/2 mode are possible because they run as a subset of the Application Program Interface (API) of OS/2 called the "family" API. [Ref. 7:p. 787] Application programs written using the family API can run on DOS or OS/2 operating systems. Both OS/2 and Windows require the 80386 CPU at a minimum.

In the OS/2 and WIN-OS/2 modes three types of multitasking are supported. In "time slice" multitasking each running process or program is given a share CPU time as established by the user (or by default). The more programs that are running the less time is available for a single program. OS/2 "preemptive" multitasking is based on a predetermined priority scheme. OS/2 supports "cooperative" multitasking for Windows applications These applications keep track of CPU time used and return control of the CPU willfully, without operating system intervention.

DOS applications do not use cooperative multitasking. OS/2 does not support the Windows 386 enhanced mode for DOS because it uses hardware specific features [Ref. 10:p. 195]. OS/2 does have various settings to allow DOS compatible programs to run as efficiently as possible [Ref. 10:p. 194].

Compatible OS/2 and Windows application programs may communicate using a productivity application called "Data Update". [Ref. 39:p. 49] This permits a dynamic data exchange (DDE) between the OS/2 and Windows programs while they are running. Normally program data is private. "Data Update" provides a link between programs.

OS/2 supports *dynamic linking* of processes located in *dynamic link libraries* (DLL's). [Ref. 7:p. 793,794] Dynamic linking allows a program to call a subroutine that has not been linked to executable machine code in memory. In static linking the external subroutine has been linked prior to loading the program in memory and prior to executing it. Dynamic linking has an advantage over static linking because it is easier to incorporate new subroutines into the DLL's. In dynamic linking, references to DLL's are resolved by the loader.

DLL's contain the OS/2 application programming interface (API). [Ref. 7:p. 793] This allows software developers to write DLL's without modifying the operating system and supports an *open system architecture*. (DLL's are also used with Windows for Workgroups [Ref. 10:p. 403].)

OS/2 standard edition is designed for the desktop environment and as a result gives the user great latitude in process management. To show the user the status of a process OS/2 provides the "PSTAT" command. [Ref. 39:p. 162] This command may be used with or without switches to display information about a process or a thread. Some of the status information available from "PSTAT" includes: DLL's, PID, shared memory, and semaphore settings.

OS/2 blends features of UNIX and Windows with features of its own. OS/2 can emulate a Windows session. OS/2 gives the user more control of processes than UNIX with commands such as: MAXWAIT, THREADS, and TIMESLICE. These commands are possible because OS/2 is a single user operating system that supports multitasking. UNIX design is for many users with many tasks. UNIX does not give the individual user as much control over processes as does OS/2.

C. OS/2 FILE SYSTEM and FILE SECURITY

When OS/2 is installed the user is given the choice of two file systems for fixed disks. One is the DOS compatible File Allocation Table (FAT) with the associated File Control Block (FCB) structure. [Ref. 10:p. 174] [Ref. 39:p. 92] The other is the OS/2 supported High Performance File System (HPFS).

The FAT was designed for relatively small file systems. The FAT contains information on every block of storage on the disk so as disk size increases the FAT gets proportionally larger. The larger the FAT the slower the file system becomes to a point of inefficiency. The FAT file system must be used for DOS and Windows application program compatibility because they cannot read an HPFS partition.

HPFS was introduced with Version 1.2 standard edition along with HPFS386 for OS/2 extended edition to support the IBM LAN server and the Microsoft LAN Manager. [Ref. 10:p. 174,184] HPFS is not compatible with DOS and Windows applications. HPFS has the ability to read the DOS FAT, but cannot read MS DOS 6.0 compressed files. HPFS is for fixed disks only and is not intended for removable disks.

HPFS eliminates the "8.3" character restriction for file name and file extension used with DOS systems. File naming under HPFS mimics Berkeley UNIX. [Ref. 10:p:175] [Ref. 39:p. 2] HPFS allows 254 characters to be used in a file name.

HPFS is case sensitive treating upper and lower case as uniquely different letters (like UNIX). Nearly all keyboard characters can be used in the file name. The four characters that are not usually permitted (left and right parenthesis, pipe and ampersand) can be used if preceded by a caret [Ref. 10:p. 176].

Certain file name extensions are reserve in OS/2. Like DOS, the OS/2 executable files end with .COM, .EXE, and .SYS. CONFIG.SYS serves the same purpose as in DOS, to tell the operating system configuration for devices. There is one exception; OS/2 uses .CMD vice .BAT for batch files [Ref. 39:p. 239].

The actual fixed disk (or a removable disk) is also called a volume. To ensure volumes do not have the same name OS/2 checksums the user supplied name with the number of seconds since 1980 (like UNIX) [Ref. 7:p. 794]. This procedure gives each volume a unique identification number.

The fixed disk can be divided into partitions. The first entry for each partition is a boot sector (or block) which is numbered zero. HPFS places a SuperBlock and a SpareBlock in block positions one and two respectively [Ref. 10:p. 177]. The SuperBlock contains pointers (addresses) to a free space bit map and bad block pointers along with a pointer to the root directory. (The SuperBlock is discussed in greater detail later, after more basic components of the file system.)

The "Fnode" is the basic allocation pointer used in HPFS. [Ref. 10:p. 177] Every file and directory is identified by an Fnode. Each Fnode uses a 512 byte block to maintain internal file information. This information includes: file attributes, file length, and the first 15 characters of the file name or directory name. Additionally, the Fnode has the access control for the file and an allocation structure that defines the actual location of the file or directory on the disk.

To speed the file retrieval process the Fnode that describes the file is located in close proximity to the file (if possible). [Ref. 10:p. 176,178] This arrangement is more flexible than the UNIX method of indirect addressing. Unix uses a sector by sector listing of files rather than the location and file length method used by OS/2. UNIX will be more efficient with small fragmented files while the OS/2 HPFS file allocation avoids the fragmentation problem and is much more efficient for retrieving large files.

The fixed disk under HPFS is divided into eight megabyte bands. [Ref. 10:p. 176] The free space bit map is located at the end of odd numbered bands and the beginning of even numbered bands. For example, free space bit maps one and two are side by side between blocks one and two. The eight megabyte bands permit a file to be stored as a contiguous 16 megabytes if needed.

The purpose of the SpareBlock is to backup write caching for error recovery and other system failures such as power loss. [Ref. 10:p. 182] [Ref. 39:p. 203,204] Write caching can be controlled in CONFIG.SYS using the "CACHE" command. The basic idea behind a write cache is to initially write data to a RAM buffer. Once the buffer is full a write is performed to the disk.

Write caching places a special demand on a disk read. When a read from a disk is requested the operating system must ensure the write to the disk has occurred. If a write failure has occurred HPFS will write the data to an emergency area. The locations of these emergency areas are kept in the SpareBlock which keeps a pointer to the emergency area directory. The OS/2 Workplace Shell GUI will notify the user when a write failure has occurred.

Before the fixed disk is read an error table in the SpareBlock called a hotfix map is consulted. [Ref. 10:p. 182] The SpareBlock also keeps a pointer to available "hotfix" repair blocks. In the event of a power failure the SpareBlock keeps a set of binary flags to determine if a normal or abnormal shut down has occurred. In the case of an abnormal shutdown OS/2 will run a check disk program (CHKDSK) and attempt to reconstruct the disk before normal system usage is resumed.

Read caching like write caching decreases the number of times a disk needs to be accessed and increases the overall system speed. [Ref. 10:p. 181] Read caching is the most common type of disk caching. It is much simpler to implement than write caching because no disk checks are required (other than bad blocks). Read caching is often used with DOS. The idea behind read caching is to read large sections of a disk at one time and store it in a RAM buffer, then read the buffer to decrease disk accesses. HPFS reads 2,048 bytes at a time into RAM. HPFS uses intelligent read caching by keeping file usage patterns in the file directory in an effort to guess what the user will request next.

To find specific files, HPFS uses directories like other systems and places them in a tree structure. [Ref. 10:p. 179] HPFS then sorts the directories by their names. This eliminates the need for a linear search, but places an additional burden on the system in keeping a balanced structure. Directories in OS/2 are two kilobytes in length. Once the root directory is full, HPFS creates another layer of directory blocks and the root directory becomes a list of directory blocks. The normal sequence to find a file starts at the SuperBlock that contains a pointer to the root directory Fnode; this has the location of the file in the path name.

Recall that DOS application programs can not use HPFS and that HPFS is for fixed disks only. Since OS/2 supports both FAT and HPFS a brief comparison is in order: [Ref. 10:p. 183,184]

* HPFS allows file names to be 254 characters while DOS allows only eleven ("8.3").
* HPFS keeps up to 64 kilobytes of file attribute information while DOS uses five bits.
* HPFS uses a sorted tree to store directories and eliminates the need for a linear search while DOS uses a unsorted linear list.
* HPFS uses the Fnode located near the physical file to keep file allocation information while DOS keeps allocation information in the FAT, which is at the beginning of the disk.
* HPFS uses five hundred twelve bytes (an Fnode) to store a directory while DOS uses thirty two bytes plus a variable length directory.
* HPFS uses an intelligent read caching and adjustable write caching while DOS uses only a simple read cache.

The majority of the basic file commands in OS/2 are the same as DOS. OS/2 2.1 and DOS 6.0 both support "RESTORE" and "UNDELETE" commands for file recovery [Ref. 24:p. 407,754] [Ref. 39:p. 170,186]. Some commands like compare (COMP) and disk compare (DISKCOMP) differ [Ref. 39:p. 96,113]. OS/2 does not permit the same syntax (switch settings) as DOS for these commands.

Because the philosophy behind OS/2 is one user doing many tasks, no special security measures are used with HPFS (or FAT). OS/2 supports a Lockup function to password-protect the computer by writing a batch file. [Ref. 39:p. 31] The user executes the batch file before leaving the computer. This function is very limited and may be bypassed by shutting the system down and entering a new password.

In the LAN environment, HPFS386 allows trusted programs to control the file system and must be used with Microsoft's LAN Manager or IBM's LAN Server networking software. [Ref. 10:p. 184] Both of these LAN programs are OS/2-based. File security is provided by server software and not by OS/2 itself.

D. OS/2 INPUT/OUTPUT SYSTEM

OS/2 uses features found in DOS and UNIX to accomplish input and output (I/O) services. OS/2 depends on device drivers to handle the specifics of controlling any I/O service. Like other operating systems, both character and block device drivers are supported by OS/2.

Character device drivers are used for serial I/O to a device like a modem or output to a printer. Block device drivers are used for devices that are randomly accessed like a disk drive. [Ref. 7:p. 808] Using the same strategy as DOS, OS/2 places device drivers in the CONFIG.SYS file that is read at start up [Ref. 39:p. 207]. This allows OS/2 to be device independent. Users may install a new device using the "DDINSTAL" command [Ref. 39:p. 102]. Like DOS, the "DEVICE =" command is placed in CONFIG.SYS. This is an advantage over UNIX systems where the system administrator (superuser) must link new devices to the operating system.

Device drivers contain device specific details of a device. These details need not be known by OS/2 or the application program requesting I/O. Being a multitasking system that also supports DOS and Windows applications, places some extra requirements on the OS/2 I/O system. When a device driver is written for OS/2, it must be written to support two modes of operation. [Ref. 7:p. 808] The drivers support single tasking with the *real mode* and multitasking with a *protected mode*. These modes parallel the modes available on the 80286 and follow-on CPU's.

Each device driver in OS/2 has two basic parts: a *device strategy routine* and a *device interrupt routine*. [Ref. 7:p. 808] The strategy routine places the I/O request in a queue and determines if the device is available and initiates the I/O. The interrupt routine will unblock any blocked process waiting for the I/O. The interrupt routine takes over to handle the I/O and lets the strategy routine know the device is once again available.

OS/2 uses background programs like UNIX daemons to monitor I/O processes. [Ref. 7:p. 804] OS/2 must not allow any single application session to crash the system. DOS handles an I/O error by stopping the entire system until the user responds. In OS/2 processes are ran in sessions or screen groups. [Ref. 7:p. 808] The active screen group may not be the cause of the I/O error. The hard error daemon in OS/2 determines which process caused the error and stops the entire system from crashing. The "AUTOFAIL" command in CONFIG.SYS give the user the option of seeing the screen where the error occurred or display of an error message [Ref. 39:p. 198].

OS/2 offers the same flexibility as DOS to install new devices and the background monitoring ability of UNIX to control the system I/O. OS/2 also supports 32 bit devices.

E. OS/2 USER INTERFACE

OS/2 supports four different user interfaces. The primary user interface called, Workplace Shell, was introduced with the release of OS/2 Version 2.0 to replace Presentation Manager. The two graphical interfaces provided are Workplace Shell and Windows [Ref. 10:p. 195]. Two separate command line interfaces are provided: one for OS/2 and one for DOS [Ref. 10:p. 193,194].

The original Presentation Manager was an extended version of Microsoft Windows that used buttons, dialog boxes, menus and tiled windows. [Ref. 7:p. 804] [Ref. 10:p. 185] The tiled windows were side-by-side (like the DOS Shell) as opposed to the overlapping windows used with OS/2's Workplace Shell. The Workplace Shell marked a trend toward an object oriented graphical user interface that resembles previous efforts of the Apple Macintosh. (OS/2 uses a shredder in place of the Macintosh trash can for deleting files.)

Objects fall into one of four categories: *data file objects, device objects, program objects* and *folder objects*. [Ref. 10:p. 185,186] [Ref. 39:p. 14] All objects are represented as icons. Data file objects are the files the programs create and use; they include text, spreadsheets, sound and video. Device objects are the physical devices. Common devices are: printers, plotters and modems. Program objects are icons to represent programs on the system like spreadsheets and word processors.

Folder objects hold a collection of other objects. A special folder, called the "Desktop", fills the screen with all the available objects. [Ref. 39:p. 15] Folders may hold folders to provide a user defined hierarchy. The primary folder is the System folder that forms the top of the hierarchy (like the Macintosh).

Since the folder is a collection of objects, OS/2 provides two views of a folder in addition to the folder icon. [Ref. 10:p. 189] [Ref. 39:p. 34] The folder may be viewed as a tree structure or a list of files in a directory style format called *details*. Files in a folder are treated as objects allowing the user to select a file icon and automatically activate a program object associated with the file (like the Macintosh).

The Workplace Shell is the default interface in OS/2. An important unique feature of the Workplace Shell is that this environment allows the user to run multiple simultaneous sessions of OS/2, DOS, and Windows. Each type of program is usually maintained in a separate folder for easy access. Starting a session is essentially the same as any other GUI. The user simply double clicks on an icon.

Borrowing some Macintosh terminology, the Workplace Shell environment is referred to as a Desktop which comes with a "clipboard" and several "Productivity Applications" (PA's).

[Ref. 39:p. 46] Workplace Shell supplies a clipboard as a common area where data between sessions can be exchanged. PA's resemble Macintosh "Desk Accessories" (DA's) with expanded functionality. OS/2 Version 2.1 provides twenty five dedicated PA's and six games [Ref. 39:p. 45-64].

Ten PA's may be used for personal planning in every possible realm. Daily and monthly plans can be tied in with alarms and later archived. PA's are also provided for: calculators, databases, FAX, graphics, text editors, spreadsheets, sounds, and system monitoring (to mention a few). Some of the popular games are: chess, jig saw, scramble, and solitaire. PA's are located in the Productivity folder.

The GUI alternative to Workplace Shell is the Windows environment. OS/2 has Windows embedded in the operating system. [Ref. 10:p. 195] [Ref. 39:p. 69] There is a restriction to running programs under the Windows environment; they must run in a standard mode because OS/2 does not support Windows enhanced mode, which is required to run programs like Microsoft PowerPoint. Any programs that use low level hardware system calls like the Windows "386" enhanced mode cannot run under OS/2.

Windows programs that are OS/2 compatible can be run one of three ways. [Ref. 39:p. 69] First, Windows may be run as a full scale Windows application. This is a default mode where Windows takes over the entire screen. Second, Windows applications may be run as a WIN-OS/2 session, where several Windows applications are active at one time. Third, Windows applications may be ran under an OS/2 session as a separate WIN-OS/2 session. This mode requires special settings to allow both OS/2 and Windows applications to be active.

Moving from the original OS/2 command line environment to a graphical user interface created a great deal of operating system memory overhead. OS/2 added 500 additional function calls to support object oriented GUI [Ref. 7:p. 804]. The function calls handle windowing and graphics programming interface (GPI). GPI is the graphics equivalent to Applications Programming Interface (API) where programmers can use operating system functions.

To remain flexible, OS/2 continues to support a command line interface of its own and one for DOS. [Ref. 10:p. 193] The command line user interface has some important functions, such as file comparison, that are not supported by Workplace Shell. Most of the commands used with the OS/2 command line are the same as the ones used with MS DOS command line. The OS/2 command line interface supports the "Restructured Extended Executor" (REXX) batch programming language.

The OS/2 Workplace Shell and the OS/2 programming language REXX are part of IBM's standardization project called System Application Architecture (SAA). [Ref. 10:p. 193] [Ref.

39:p. 157] The primary goal of SAA is to present a common interface across many systems. A user familiar with the SAA environment on a PC will have the same interface with an IBM mid-size or mainframe computer. This way the user does not have to relearn a new system with every migration.

SAA strives for commonalities in three areas: communications support, programming interface, and user access [Ref. 10:p. 187]. Common communications support is accomplished by IBM's Extended Services (see OS/2 Networking). The Workplace Shell supports common user access and REXX supports the common programming interface.

F. OS/2 Networking

OS/2 supports communication and data base management in its Extended Edition add on packages. These features are not part of Standard Edition OS/2. With the release of OS/2 Standard Edition Version 2.0 the networking package received a new name christened "Extended Services" Version 1.0 [Ref. 41:p. 39]. The package contains Communications Manager which supports a wide variety of options.

Features of the Extended Services include: [Ref. 10:p. 197,198] [Ref. 41:p. 39-43]

* *Integrated terminal emulation capabilities.* Provides terminal emulation capabilities for plain ASCII, IBM 3270, and IBM 5250 terminals. It enables file transfers to take place between the PC and more than one host system at a time. As many as five AS/400 System/36 terminal or printer sessions can be run at the same time under 5250 emulation. (AS/400 is IBM's mid-size computer system).
* *Network Device Interface Specification* (NDIS) support. An emerging standard for low level interfaces to network interface cards. Many vendors are writing NDIS drivers for their interface cards; this gives a wide selection of network interface cards for the PC.
* *Advanced Program to Program Communications* (APPC). Provides a structure for application programs to communicate on a peer to peer basis across different

networks and hardware types.

* *Common Programming Interface for Communications* (CPI/C). Provides a common programming interface for network application programmers across a broad range of System Application Architecture (SAA) compliant IBM hardware and software.

* *3174 Peer Communications Support.* Allows workstations that use coaxial cable connections (not connected to a LAN), to share files. Also enables APPC applications to connect to peer applications on host systems or other workstations.

* *SNA LAN gateway support.* Enables a single OS/2 workstation to act as a host gateway simultaneously for as many as 127 other workstations on a local area network.

* *X.25 Protocol Support.* Enables the workstation to communicate over this very popular wide area network protocol that is supported by the International Standards Organization (ISO).

* *System/370 NetView Support.* Enables network administrators to use NetView to issue OS/2 commands on remote workstations and servers for network management.

* *Logical Unit Application* (LUA) interfaces. Supports Logical Unit sessions under IBM's SNA.

* *Asynchronous Communications Device Interface* (ACDI). Provides a programming interface to asynchronous

communications devices. This isolates the application program from the specifics of a particular device.

* *LAN Interfaces*. Many LAN operating systems support OS/2 servers and workstations, including: Novell NetWare, IBM LAN Server, Microsoft LAN Manager, Banyan VINES, and TCP/IP for OS/2 servers. IBM LAN Requester is used by OS/2 and DOS workstations.

OS/2 is the base operating system that both LAN Manager and LAN Server are built upon. OS/2 Version 2.0 and above are not compatible with LAN Manager since it is based on Version 1.3. These versions will have to use LAN Server software.

In short, OS/2 is supported in nearly every networking environment under the Extended Services 1.0. Additionally, the Standard Edition of OS/2 Version 2.1 has extensive groupware capabilities. [Ref. 42:p. 34-36] It can run Windows for WorkGroups under the Windows emulation mode and Lotus Notes groupware built for OS/2. WordPerfect Office for OS/2 groupware is scheduled for release in the first quarter of 1994.

G. OS/2 ADVANTAGES and DISADVANTAGES

OS/2 was the first PC operating system built from the ground up to support a 32 bit architecture and multitasking. [Ref. 10:p. 168] [Ref. 39:p. 3] Minimum hardware requirements include: the 80386 microprocessor, a 60 megabyte fixed disk drive with 16 to 30 megabytes available for OS/2, and a minimum of four megabytes of RAM. (For good performance eight megabytes are required to run multiple sessions effectively.) Additionally, a high density removable disk drive supporting disk capacities over one megabyte is required.

Compared to DOS alone these hardware requirements are quite extensive, but DOS does not have the multitasking capability of OS/2. When DOS is teamed with Windows the hardware requirements are nearly the same as OS/2. *OS/2 is the only PC operating system under 30 megabytes in size that provides 32 bit preemptive multitasking.*

MS DOS and Windows have a much larger market share than OS/2. Because DOS and Windows have the greatest market share, software developers normally write application programs for those products first and later migrate the products to OS/2 and UNIX. OS/2 is less expensive than DOS and Windows combined. OS/2 supports "most" DOS and Windows application programs. OS/2 will not support DOS and Windows applications that use low level hardware functions such as the Windows "386" enhanced mode. OS/2 has an overall advantage over the

DOS and Windows combination in cost and functionality.

Another major competitor to the OS/2 market is the Apple Macintosh with the System 7 operating system. System 7 is available on Apple Macintosh products only. Unlike OS/2, Macintosh requires additional hardware to run DOS products. This is because Macintosh and PC hardware have significant differences. (Some Macintosh product lines like the Quadra, can read and write DOS diskettes without add-on hardware.) OS/2 does not support Apple software directly.

Users see definite similarities between the GUI's of OS/2, System 7, and Windows, but looks can be deceiving. System 7 is part of an integrated package that comes bundled in ROM boards inside the Macintosh. System 7 and Windows basic functions are similar to OS/2, but they *not* a full 32 bit architecture like OS/2. System 7 does offer file security not found in OS/2 or Windows.

When OS/2 is compared to other 32 bit operating systems, like UNIX and WindowsNT, hardware requirements for OS/2 are reasonable. UNIX will vary greatly in size depending on which proprietary version is used on a particular computer. UNIX can run on an 80386 like OS/2, but the memory requirements for UNIX are more than twice as great as OS/2.

Unlike UNIX, OS/2 and WindowsNT are one company products that have a relatively fixed memory requirement. WindowsNT requires much more memory and processing power than OS/2.

WindowsNT running on a Digital Equipment Corporation (DEC) Alpha chip (the world's fastest at 150 Mhz) requires 16 megabytes of RAM and 90 megabytes of fixed disk space. WindowsNT running on the considerably slower Intel 80486DX (66 Mhz) requires 12 megabytes of RAM and 70 megabytes of fixed disk space.

Comparing OS/2 with UNIX and WindowsNT may not be quite fair because both cost more and both are much larger operating systems. The design of OS/2 was for one user doing many tasks. Both UNIX and WindowsNT are designed for many users doing many tasks. One definite advantage of UNIX and WindowsNT are C2 security measures that are not offered with OS/2.

Slow acceptance has been another major disadvantage for OS/2. A bad reputation developed when new releases of OS/2 were always late and lacked promised functionality. The functionality issues seem to have been cured with Versions 2.0 and 2.1; however, both arrived in the market later than expected.

The OS/2 Workplace Shell is meeting with some resistance because it differs from the Windows environment. The user can run Windows applications from the WIN-OS/2 or Workplace Shell. The Workplace Shell is geared to an object oriented environment which is consistent with programming paradigms used with the C++ programming language.

Once the user becomes familiar with the Workplace Shell, the knowledge gained can be transferred to other IBM platforms consistent with IBM's Systems Application Architecture (SAA). IBM has included OS/2 in the System Network Architecture (SNA) and other protocol suites.

OS/2 Extended Services 1.0 is supported by major network protocols. In the LAN environment the OS/2 Extended Services supports several LAN servers such as IBM LAN Server, Microsoft LAN Manager and Novell Netware.

OS/2 Standard Edition 2.1 can use groupware such as Workgroups for Windows (WFW), and Lotus Notes. WordPerfect Office groupware for OS/2 is expected to be released in 1994.

For users not needing networking capabilities the Standard Edition meets single use requirements in a 32 bit multitasking environment with reasonable memory requirements and at a reasonable price. IBM continues to provide price incentives to convert users to OS/2.

VIII. OPERATING SYSTEMS SUMMARY

A. INTRODUCTION

The selection of an operating system for PC's or workstations that are members of a network should be based on compatibility, cost, and performance. Compatibility is especially important because there is a great variety of hardware and software in a network. In a network, the PC or workstation operating system must work with the network operating system. Both of these operating systems must work with system software, like memory managers, and application program software. An incompatibility of any pair of these resources can render a computer or the entire network inoperative. Other factors such as support for input/output devices and overall speed of the operating system should also be considered.

Many times the hardware selection will influence what operating system is included. Macintosh users will receive System 7 by default. Powerful workstation users traditionally use UNIX. IBM PC users may use IBM PC DOS or OS/2 supplied by IBM. Many IBM clone PC's have MS DOS with Windows pre-installed. The PC or workstation operating system will influence the network operating system selection and the availability of application software.

This summary emphasizes hardware requirements of PC operating systems first and then reviews capabilities of the operating systems. UNIX presents a unique dilemma since it has so many proprietary versions. The UNIX model selected is the AT&T version of System V Release 4 (SVR4) which has been adopted by the SunOS in their Solaris 2.x operating system.

Some new material is presented which includes: Apple's System 7 for the Macintosh, IBM's LAN Server, and two Microsoft products, LAN Manager and WindowsNT. These systems are an integral part of the PC operating systems in the client-server networking environment. Each section is discussed and then summarized in a table format where applicable.

B. CPU REQUIREMENTS

IBM and Microsoft have traditionally based their operating systems on Intel-based microprocessors. Apple has always used the Motorola 68000 series for their computers. Varieties of UNIX can be found on both Intel and Motorola CPU platforms. Apple's A/UX is a Motorola-based product, while the SunOS is an Intel-based example of UNIX. The Macintosh Quadra may have both Motorola and Intel chips installed on the Houdini board [Ref. 43:p 4F] [Ref. 44:p. 26].

Looking at the low-end requirements for CPU's, consideration must be given to an acceptable performance. For example, Microsoft claims that Windows products can run on an Intel 80286 [Ref. 45:p. 48]. This may be true, but the performance is so poor that the run becomes a very slow walk. Windows 3.1 needs an Intel 80386 to run effectively. IBM recommends the Intel 80386 as a minimum after finding that OS/2 simply would not work on the 80286. Solaris on the Spark workstation may run several 80386's in parallel under the direction of the SunOS (Solaris) operating system. Both LAN Manager and LAN Server need an 80386 to be effective.

Most PC's in use today are running the 80286 or faster CPU's. DOS has maintained a backward compatibility to 8086 and 8088 CPU's. Microsoft WindowsNT requires the Intel 80486 for acceptable performance. WindowsNT is best served by faster CPU's like the DEC Alpha chip. The following table shows operating systems and networking systems with what should be considered the low-end microprocessor requirement for acceptable system performance.

TABLE 3 MINIMUM CPU REQUIREMENTS

System	CPU
MS DOS	8086/8088
OS/2	80386
System 7	M68030
UNIX	80386
LAN Manager	80386
LAN Server	80386
WindowsNT	80486

All newer operating systems require an 80386 or faster CPU to run efficiently. (The Motorola 68030 is roughly equivalent to the 80386.) Intel and Motorola are both introducing new CPU's. Microsoft is developing software for Intel's Pentium. Apple and IBM are developing for Motorola's PowerPC chip. Pentium and PowerPC are the replacements for the 80486 and M68040 respectively.

C. MEMORY REQUIREMENTS

Operating systems have two memory requirements, fixed disk storage and RAM. After the operating system is installed, fixed disk space must be allocated for application programs. The amount of RAM available affects system performance. A shortage of RAM causes the application to run slow or not run at all.

Fixed disk space requirements for operating systems are variable. The entire operating system need not be loaded on fixed disk if certain functions are not used. For example, OS/2 can vary in size from 16 to 30 megabytes depending on what features are installed. MS DOS 6.2 requires eight megabytes when fully installed. DOS may also be installed with or without some of the available utilities.

Compression programs like DoubleSpace or Stacker can increase available fixed disk space once installed. Users are cautioned about compression programs because they can slow system performance. OS/2 and WindowsNT cannot read MS DOS compressed files.

MS DOS 6.2, Windows 3.1, and System 7 can all be accommodated on a 20 megabyte fixed disk [Ref 44:p. 23]. OS/2 can vary in size up to 30 megabytes, so a 40 megabyte disk is considered reasonable to house the operating system, a few application programs and user files. Minimal installation of WindowsNT ranges in size from 75 to 90 megabytes [Ref. 46:p.

4] WindowsNT is in the same category as UNIX Solaris, which has a recommended fixed disk space of 200 megabytes [Ref. 12:p. 12].

RAM presents a different sort of problem. DOS and one application can run in as little as 512K of RAM. Once Windows 3.1 is added to DOS the RAM requirement increases to four megabytes, the same as OS/2. Macintosh System 7 requires a minimum of three megabytes of RAM [Ref. 10:p 249]. These RAM requirements should be considered minimums. For good system response, users need about twice the minimum RAM requirement. This translates to about eight megabytes for OS/2 and Windows and about six megabytes for System 7. Unlike the other operating systems, at least one third of System 7 is based in ROM.

The minimum RAM required for Solaris (80386 and SPARC workstations) is 12 megabytes with 16 megabytes recommended. [Ref. 12:p. 12] This is the same minimum RAM requirement that exists for WindowsNT.

TABLE 4 lists the minimal (MIN) and recommended (REC) RAM requirements for each system. This is followed by a general category of fixed disk space required for each system. "FIXED" is the fixed disk space rounded up to the closest fixed disk size. "K" is used for kilobytes and "M" for megabytes.

TABLE 4 MEMORY REQUIREMENTS

System	RAM (MIN)	RAM (REC)	FIXED
MS DOS	512K	1M	20M
OS/2	4M	8M	40M
System 7	3M	6M	20M
UNIX	12M	16M	200M
WindowsNT	12M	16M	200M

The Windows operating system environment is not shown in the table because it is not a stand-alone operating system. Windows 3.1 with DOS has the same memory requirements as OS/2.

D. PROCESS MANAGEMENT

The way a computer handles processes is also an important consideration. Multitasking capabilities are important for some users. Users may want to accomplish several tasks concurrently. The 32 bit operating systems: OS/2, UNIX, and WindowsNT offer preemptive multitasking.

The 16 bit operating systems, DOS with Windows 3.1 and System 7, perform cooperative multitasking through special application programs. They also permit the user to assign memory areas and percentages of CPU time to permit a time slice multitasking environment. DeskView is an example of third party software that performs the same function as Windows. MS DOS by itself performs no multitasking.

TABLE 5 is a summary of operating system multitasking categorized by preemptive and non-preemptive multitasking.

TABLE 5 MULTITASKING

Preemptive	**Non-Preemptive**
OS/2	Windows
UNIX	System 7
WindowsNT	DeskView

E. FILE SYSTEMS

All file systems that have been discussed have compatibility with the most widely used file system in the world, the DOS File Allocation Table (FAT). SuperDrive and Apple File Exchange program permit the Macintosh to translate and read DOS files in the Macintosh Hierarchical File System (HFS). [Ref 10:p. 266] Other systems like OS/2, allow the user to set up the fixed disk as a FAT file system or a High Performance File System (HPFS). Since LAN Manager and LAM Server are OS/2 based they support the same file systems as

OS/2. DOS cannot read HPFS.

Solaris has made great strides in compatibility with AT&T SVR4 and Berkeley UNIX File System (UFS). [Ref. 12:p 18] Solaris supports the "fdisk" utility for DOS FAT compatibility.

DOS and Windows recognize the FAT only. The DOS FAT becomes inefficient as it increases in size due to a linear search for files. This problem is corrected with HPFS which sorts files and uses the fnode for location. HPFS works very well with large files where UFS is less efficient with large files but more efficient with small fragmented files.

WindowsNT allows the choice of several file systems. WindowsNT recognizes nearly all file systems except UNIX, unless the system is exactly POSIX compliant. [Ref. 47:p. 3] WindowsNT native file system is called NTFS.

TABLE 6 summarizes operating systems and the file systems they support. Recall that System 7 requires additional hardware and software to translate and read the DOS FAT.

TABLE 6 FILE SYSTEMS

System	File System(s)
MS DOS	FAT
OS/2	FAT, HPFS
System 7	FAT, HFS
UNIX	FAT, UFS
WindowsNT	FAT, HPFS, NTFS
LAN Manager	FAT, HPFS386
LAN Server	FAT, HPFS386

F. INPUT/OUTPUT SYSTEMS

The input and output characteristics of an operating system deals with the flexibility and ease of installing new devices. Device-specific drivers interface hardware to the operating system. Because DOS is the most widely used operating system it has the greatest device support. Similar to DOS, OS/2 and WindowsNT device drivers are placed in the CONFIG.SYS file. Device drivers may be installed using the "DEVICE=" command.

System 7 controls devices through the Chooser subsystem. [Ref. 10:p. 256] Control panels give users access to device drivers that Macintosh calls Cdev's (Control Devices) or Chooser Extensions.

UNIX handles device drivers differently from DOS and System 7. By convention, UNIX device drivers are placed in a directory called "/dev". The system administrator must link the device drivers to the operating system kernel. This is a more complex task than the methods used by DOS and System 7.

The TABLE 7 outlines the operating systems and where the device drivers are stored.

TABLE 7 DEVICE DRIVERS

System	**Driver Location**
DOS	CONFIG.SYS
System 7	Chooser
UNIX	/dev

Note that CONFIG.SYS is used not only by DOS, but also by OS/2, LAN Manager, LAN Server, and WindowsNT. DOS device drivers need not be linked to the system like UNIX.

G. NETWORKING

All of the operating systems discussed here are supported by the TCP/IP protocol. OS/2 Extended Services is geared toward the IBM SNA environment. Macintosh networks can use LocalTalk, EtherTalk, or TokenTalk [Ref. 44:p. 380].

LAN's are supported by AppleShare, LAN Manager, LAN Server, UNIX and WindowsNT in a client server environment. The idea of a zero slot LAN, meaning a network interface card is not required, is supported by MS DOS 6.0 INTERLINK and System 7 using AFP (AppleTalk Filing Protocol). [Ref. 10:p. 257] DOS and System 7 can communicate with like systems using serial ports.

A strong point of Microsoft's WindowsNT is that this operating system can be networked in all environments. [Ref. 47:p. 5] WindowsNT comes bundled with TCP/IP and NetBUI protocols. Additional Microsoft networking software includes Windows for Workgroups (WFW) and Workgroup Connection. WFW is an application program that runs on top of DOS to provide a peer to peer environment. Workgroup Connection is a DOS application that allows MS DOS 6.0 (and above) users to communicate with WFW users without the Windows overhead.

E. APPLICATION PROGRAM SUPPORT

The job of the operating system is to provide an environment to support application programs. [Ref. 10:p. 907] This is accomplished by vendors publishing Application Program Interfaces (API's). API's are sometimes referred to as system calls or as hooks. API's are used by programmers to invoke functions from the operating system to support applications. This allows third party vendors to write application programs for a particular operating system.

DOS has, by far, the largest base of application programs at the lowest cost. OS/2 and WindowsNT users can take advantage of most DOS and Windows applications. Macintosh can run some DOS applications by emulation using SoftPC.

Some DOS and Windows programs that bypass the operating system and make hardware interrupt calls directly will only run in a native (DOS or Windows) environment. The programs that make these low-level calls are not supported by other operating systems.

An application developed for DOS is normally less expensive than its adaptation for another operating system. This is especially true when the adaptation is for a GUI environment. For example, Microsoft will normally develop an application for DOS first then adapt the program for Windows and Macintosh users. The adapted applications for the GUI environment are almost always more expensive. UNIX applications are usually the most expensive because each version of UNIX may need the application adapted to it.

OS/2, UNIX, and WindowsNT can run 32 bit applications. The total number of 32 bit applications is quite small when compared to the number of 16 bit applications available. WindowNT has the ability to run OS/2 and POSIX compliant UNIX applications. Neither OS/2 nor UNIX can run applications designed specifically for WindowsNT.

I. CONCLUSIONS

The emphasis here has been PC and workstation operating systems. Operating systems are built for different operating environments. UNIX is designed for many users performing many tasks concurrently. DOS is designed for one user performing one task at a time. OS/2 is designed for one user performing many tasks. The other PC operating system mentioned, but not covered in detail, was System 7.

Apple introduced System 7 with the Macintosh as part of an integrated package of hardware and software. This type of system is called "plug and go", which means the user has all of the hardware and software together in one package. The user friendliness of the Macintosh made it very popular for home computing.

LIST OF REFERENCES

1. Long, Larry and Long, Nancy. Computers, Second Edition. Prentice Hall, 1990.

2. Luce, Thom. Computer Hardware, System Software, and Architecture. Mitchell Publishing Inc., 1989.

3. Schuytema, Paul C., "Infinite Expansion". **Compute** Magazine; November 1993.

4. Methvin, Dave, "CPU Line Expands to Meet Diverse Uses". **PC Week** Magazine; November 1993.

5. Gookin, Dan, "The Ultimate Windows Machine". **Compute** Magazine; February 1993.

6. Seymour, Jim, "Pentium: The Second Wave". **PC** Magazine; 25 January 1994.

7. Deitel, H.M., Operating Systems, Second Edition. Addison Wesley Publishing Company, 1990.

8. Lohman, Major E.A., "Warchest Demos CD-ROM's Suitability for Battlefield information". **Vizions** Magazine; Fall 1991.

9. Stamper, David. Business Data Communications, Third Edition. The Benjamin/Cummings Publishing Company Inc., 1991.

10. Day, Michael et. al. LAN Operating Systems. New Riders Publishing, 1993.

11. Tanenbaum, Andrew S. Modern Operating Systems. Prentice Hall, 1992.

12. Solaris 2.x for .86 Platforms **White Paper** by SunSoft (A Sun Microsystems Business).

13. Open Systems Environment (OSE) Profile for Imminent Acquisitions. (Part 8 of 10 parts) Operating Systems Department of Defense Draft.

14. Kelly-Bootle, Stan. Understanding UNIX. Sybex Inc., 1993.

15. Federal Information Processing Standard. by U.S. Department of Commerce; May 1993.

16. Microsoft MS-DOS Operating System 5.0 User's Guide and Reference. Microsoft Corporation. 1992.

17. Minasi, Mark. "Productivity Choice: MS-DOS 6.0". **Compute** Magazine; September 1993.

18. Minasi, Mark. "Dos 6.0: SOS Reaches a New level". **Compute Magazine**; June 1993.

19. Prosise, Jeff. "MS-DOS Addresses DoubleSpace Concerns". PC Magazine; 11 January 1994.

20. Meldelson, Edward. "IBM's PC DOS 6.1: Has MS-DOS Met Its Match". PC Magazine; 28 September 1993.

21. "Bitter Divorce Frees Microsoft, IBM, but What About the Kids?" by Reuters (Seattle) **San Jose Mercury News**; 17 September 1993.

22. Minasi, Mark. "DOS NT: A Plea". **Compute** Magazine; May 1993.

23. "Next Version of Windows Signals that Microsoft is Willing o Change". By Robin Stacy. **San Jose Mercury News**; 26 December 1993.

24. Robins, Judd. Mastering DOS 6.0 Special Edition. Sybex, Inc.,1991.

25. Bixby, Robert. "The Computer Choice Awards". **Compute** Magazine; March 1993.

26. Roberts, Tony. "Windows Users Still Need DOS". **Compute** Magazine; March 1993.

27. "Microsoft Windows Advertizement". **San Jose Mercury News**; 2 October 1993.

28. Semich, William. "NT: Is It Ready for Critical Apps". **Datamation** Magazine; 15 March, 1993.

29 Shannon, L.R., "More and More PC Users are Opening Windows". **Compute** Magazine; March 1993.

30. "Mastering WordPerfect 5.1" By Allen Simpson. **San Jose Mercury** News; 21 November 1993.

31. Prosise, Jeff. "DOS 6.0's DoubleSpace: Is It Safe?". PC Magazine; 23 November 1993.

32. Roberts, Tony. "Recovering Deleted Files". **Compute** Magazine; October 1993.

33. Prosise, Jeff. "Maximizing Memory Under DOS 6.0" PC Magazine; August 1993.

34. "Memory Managers Beg, Borrow and Steal to Boost RAM". by Philip Robinson. **San Jose Mercury News**; 16 January 1994.

35. Roberts, Tony. "Defragmenting Your Hard Disk". **Compute Magazine**; November 1993.

36. Glitman, Russell. "Novell Teaches DOS New Tricks". **PC World Magazine**; October 1993.

37. Derfler, Frank J. and Rigney, Steve. "Windows for Workgroups 3.11: The Best for All?". **PC Magazine**; 11 January 1994.

38. Dvorak, John C., "OS/2 Then and Now". **PC Magazine**; 14 September 1993.

39. Dyson, Peter. OS/2 2.1 Instant Reference. Sybex, Inc.; 1993.

40. "The OS/2 100". IBM Advertizing Supplement; 1993.

41. Shaddrick, Kelley. "Communicating Without Magic" **OS/2 Professional**; May 1993.

42. Rash, Wayne Jr., "Groping for Groupware for OS/2". **OS/2 Professional**; September 1993.

43. "The Evolution of a Revolution". by Laurie Flynn. **San Jose Mercury** News; 23 January 1993.

44. McClennand, Deke. Macintosh System 7.1: Everything You Need to Know. Sybex Inc.; 1992.

45. Microsoft LAN Manager. **Strategic White Paper**; 1992

46. "Microsoft WindowsNT Overview". Microsoft Promotional Advertizement; September 1993.

47 "WindowsNT and OS/2: The Advantages of WindowsNT for Today's Client-Server Computing". Microsoft Corporation; July 1993.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5101	2
3.	Norman Schneidewind, Code SM/SS Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Myung Suh, Code SM/SU Naval Postgraduate School Monterey, CA 93943-5000	1
5.	Lt. Charles E. Frame 6997 14TH Avenue North St. Petersburg, FL 33710	2